Classification based on dispersed data with deep learning issues

Doctoral Dissertation

Kwabena Frimpong Marfo

Supervised by:

dr hab. Małgorzata Przybyła-Kasperek, prof. UŚ

University of Silesia in Katowice

Institute of Computer Science

2025

Contents

Abstract

This thesis presents a series of publications that addresses the challenge of constructing optimal global classification models from dispersed and heterogeneous datasets without requiring centralized data access. Here, dispersed or distributed data refers to datasets stored across multiple independent sources that cannot be physically consolidated, while heterogeneous data denotes collections whose feature spaces and object distributions differ across locations. Traditional approaches to dispersed data environments assume uniform feature spaces and direct data aggregation, making them unsuitable for real-world scenarios involving privacy constraints and structural data variability. To overcome these limitations, this research introduces and evaluates three complementary neural network approaches, each designed to address the challenges of non-homogeneous and distributed data. The first approach integrates predictions from locally trained k-nearest neighbors classifiers through a neural network that performs a fusion of independent outputs, achieving a unified global classification decision while preserving data privacy. The second approach employs an artificial-object imputation mechanism that harmonizes structural differences among local datasets, enabling aggregation of the trained network weights into a single global model. Lastly, the third approach applies feature-extraction techniques such as Principal Component Analysis (PCA), Singular Value Decomposition (SVD) and Uniform Manifold Approximation and Projection (UMAP) to project local datasets into a unified feature space, after which the resulting models are combined using a soft voting scheme. Experimental evaluations demonstrate that these approaches consistently enhance classification accuracy and robustness compared to classical ensemble and centralized methods, while ensuring privacy preservation and computational efficiency. The findings contribute novel insights into neural network-based distributed learning and establishes a foundation for scalable, privacy-aware data integration in dispersed data environments.

List of publications

[**P1**] Przybyła-Kasperek M., Marfo K.F. Neural network used for the fusion of predictions obtained by the k-nearest neighbors algorithm based on independent data sources. *Entropy*, 23(12), 2021. doi:10.3390/e23121568
URL: http://dx.doi.org/10.3390/e23121568
DOI: 10.3390/e23121568
$\text{MEiN}_{2021} = 100$
Number of citations:

- according to Web of Science: 11
- according to Google Scholar: 16

[**P2**] Przybyła-Kasperek M., Marfo K.F. Influence of noise and data characteristics on classification quality of dispersed data using neural networks on the fusion of predictions *International Conference Information Systems Development*, 2022. doi: 10.62036/ISD.2022
URL: http://dx.doi.org/10.62036/ISD.2022
DOI: 10.62036/ISD.2022
$\text{MEiN}_{2022} = 140$
Number of citations:

- according to Web of Science: 0
- according to Google Scholar: 3

[**P3**] Marfo K.F., Przybyła-Kasperek M. Radial basis function network for aggregating predictions of k-nearest neighbors local models generated based on independent data sets *Procedia Computer Science*, 207:3234–3243, 2022.
URL: http://dx.doi.org/10.1016/j.procs.2022.09.381
DOI: 10.1016/j.procs.2022.09.381
$\text{MEiN}_{2022} = 70$
Number of citations:

- according to Web of Science: 0
- according to Google Scholar: 8

[**P4**] Marfo K.F., Przybyła-Kasperek M. Radial basis function neural network with a centers training stage for prediction based on dispersed image data *International Conference on Computational Science*, 10476:89–103, 2023.
URL: https://link.springer.com/chapter/10.1007/978-3-031-36027-5_7
DOI: 10.1007/978-3-031-36027-5_7
$\text{MEiN}_{2023} = 140$
Number of citations:

- according to Web of Science: 0
- according to Google Scholar: 4

[**P5**] Marfo K.F., Przybyła-Kasperek M. Study on the Use of Artificially Generated Objects in the Process of Training MLP Neural Networks Based on Dispersed Data *Entropy*, 25(5), 2023.
URL: https://doi.org/10.3390/e25050703
DOI: 10.3390/e25050703
$\text{MEiN}_{2023} = 100$
Number of citations:

- according to Web of Science: 1
- according to Google Scholar: 1

[**P6**] Marfo K.F.,Przybyła-Kasperek M., Sulikowski P. Fragmented Image Classification Using Local and Global Neural Networks: Investigating the Impact of the Quantity of Artificial Objects on Model Performance *International Conference on Computational Science*, 14838:280-194, 2024.
URL: https://link.springer.com/chapter/10.1007/978-3-031-63783-4_21
DOI: 10.1007/978-3-031-63783-4_21
$\text{MEiN}_{2024} = 140$
Number of citations:

- according to Web of Science: 0
- according to Google Scholar: 0

[**P7**] Marfo K.F.,Przybyła-Kasperek M. Exploring the Impact of Object Diversity on Classification Quality in Dispersed Data Environments. *ACIIDS*, 14796:250-262, 2024.
URL: https://link.springer.com/chapter/10.1007/978-981-97-4985-0_20
DOI: 10.1007/978-981-97-4985-0_20
$\text{MEiN}_{2024} = 70$
Number of citations:

- according to Web of Science: 1
- according to Google Scholar: 2

[**P8**] Marfo K.F.,Przybyła-Kasperek M. Enhancing Dispersed Data Classification: A Hierarchical Model Based on Neural Networks *Proceedings of the 5th Polish Conference on Artificial Intelligence*, 154-161, 2024.
URL: https://pages.mini.pw.edu.pl/ estatic/pliki/PP-RAI_2024_proceedings.pdf
DOI: 10.17388/WUT.2024.0002.MiNI
$\text{MEiN}_{2024} = 20$
Number of citations:

- according to Web of Science: 0
- according to Google Scholar: 0

[**P9**] Przybyła-Kasperek M.,Marfo K.F. A multi-layer perceptron neural network for varied conditional attributes in tabular dispersed data *PLoS ONE*, 19:1-56, 2024.
URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0311041
DOI: 10.1371/journal.pone.0311041
$\text{MEiN}_{2024} = 100$
Number of citations:

- according to Web of Science: 3
- according to Google Scholar: 3

[**P10**] Marfo K.F.,Przybyła-Kasperek M. Objects Diversity and its Impact on Classification Quality in Dispersed Data Environments *Vietnam Journal of Computer Science*, SN: 2196-8888 p253, 2025.
URL: https://doi.org/10.1142/S2196888824500180
DOI: 10.1142/s2196888824500180
$\text{MEiN}_{2025} = 20$
Number of citations:

- according to Web of Science: 0
- according to Google Scholar: 0

[**P11**] Marfo K.F., Przybyła-Kasperek M., Decentralized Neural Network Modeling from Heterogeneous Data Sources: A Feature Mapping Approach *International Conference Information Systems Development*, 2025. doi: 10.62036/ISD.2025
URL: http://dx.doi.org/10.62036/ISD.2025
DOI: 10.62036/ISD.2025
$\text{MEiN}_{2025} = 140$
Number of citations:

- according to Web of Science: 0
- according to Google Scholar: 0

Chapter 1

Introduction

In this era of big data, the utilization of machine learning hinges on the accessibility and quality of training data. Yet, in many real-world applications, data are not centralized but rather dispersed across multiple sources, often stored in heterogeneous formats with varying attribute and under strict privacy constraints like the General Data Protection Regulation (GDPR) [2] and the Health Insurance Portability and Accountability Act (HIPAA) [1] laws. Dispersed data poses unique challenges for traditional machine learning models, which typically assume access to a unified and well-curated dataset stored on some central server.

To illustrate the significance of this problem, consider an example in the healthcare sector: patient data are often distributed across numerous hospitals and healthcare facilities, each utilizing distinct data management systems. These systems may vary in the structure and semantics of stored information – some may record patient vitals while others may emphasize medication histories or laboratory test results. Moreover, data protection regulations such as the GDPR in the European Union and the HIPAA in the United States, impose legal limitations on direct data sharing across institutions. Consequently, developing a global predictive model such as one for identifying patients at high risk of sepsis becomes a formidable task when data cannot be pooled or normalized.

Two existing paradigms attempt to address these issues: federated learning and distributed learning. Federated learning [36] emphasizes on decentralized training by sharing model parameters rather than raw data. Here, a central model is initialized and broadcast to local nodes where training occurs on independent datasets. Locally trained parameters are then aggregated centrally, and the cycle repeats until a stopping criterion is achieved. While this method preserves data privacy, it assumes that each local dataset has an identical feature space – an assumption often violated in practice. Distributed learning [10], on the other hand, assumes full access to data at a central location but subdivides it into chunks for parallel processing or model enhancement. This approach is impractical in scenarios constrained by privacy regulations and institutional silos.

In contrast, the research in this thesis proposes alternative approaches that assumes nei-

ther feature-space homogeneity nor centralized access. Instead, it introduces methods to construct global neural network models using local datasets that vary arbitrarily in structure – allowing for differences in both conditional attributes and object instances. Particularly, novel strategies are proposed and evaluated across three levels. The first level is the approach of the fusion of local predictions. Here, each local dataset trains a k-nearest neighbors (k-NN) classifier, and the resulting predictions are aggregated as inputs to a neural network for the final decision. This offers the advantage of preserving data privacy as well as generating unambiguous classification decisions. The second level overcomes the over-reliance on the k-NN predictions in the first level by employing a distributed neural network training and aggregation strategy, wherein structural homogeneity across local datasets is achieved by imputing missing attributes with artificially generated objects. A global model is then constructed by aggregating the weights and biases of neural networks trained on the imputed local datasets. This approach also preserves data privacy. Lastly, the third level addresses the computational overhead of artificial object generation by adopting a feature extraction-based modeling approach. Each local dataset is mapped to a unified k-dimensional feature space via feature extraction methods (e.g. PCA[34], SVD[23], UMAP[35]), ensuring structural homogeneity. After, separate neural network models are trained on these transformed datasets and their predictions are aggregated using a soft voting scheme. This approach is robust to data heterogeneity, preserves data privacy as well as simplifies the network design by standardizing input representations

1.1 Hypothesis of the doctoral thesis and contributions

The central hypothesis of this research is that, the use of neural network-based architectures across the three novel approaches described above, significantly improves the accuracy and robustness of classification in dispersed data environments. Unlike classical approaches such as centralized learning, which require all local datasets to be aggregated into one centralized dataset as well as feature-space homogeneity, or ensemble-based approaches which are sensitive to data imbalance and assumes feature-space homogeneity, the proposed methods construct global neural network models from arbitrarily structured local datasets, ensuring both data privacy and high predictive performance. On this premise, the strategies and approaches proposed are to tackle three main research questions:

1. Does the use of a neural network as a fusion method generate satisfactory classification accuracy? If yes, would it be better to use the Multi Layer Perceptron (MLP) or the Radial Basis Function (RBF) neural network?

2. Can the strategy of imputing missing attributes across local datasets and subsequently constructing a global neural network model from the aggregated trained weights improve the accuracy of classification?

3. Can the strategy of constructing a global model from the extracted features of local datasets using transformation maps such as PCA, SVD and UMAP be a more precise approach for generating satisfactory classification accuracy?

In Chapter 3, it is demonstrated that papers [**P1**], [**P2**], [**P3**] and [**P4**] address the first research question, with papers [**P7**] and [**P10**], reporting on the robustness of the approach reported in [**P3**], whereas papers [**P5**], [**P6**] and [**P9**] pertain to the second research question. Finally, papers [**P8**] and [**P11**] are shown to address the third research question. The remainder of this doctoral thesis is organized in the following way: Chapter 2 discusses the concepts of dispersed data together with methods and their applications. Chapter 3 describes proposed algorithms to overcome the challenges of dispersed data-based classification problems. Chapter 4 summarizes the thesis.

Chapter 2

Dispersed Data

This chapter describes dispersed data, explains basic theory, introduces the most popular algorithms in literature that has been used to handle problems involving dispersed data.

2.1 Dispersed Data

A dispersed data environment is a collection of independent decision tables or datasets:

$$\mathcal{D} = \{D_i = (U_i, A_i, d) \mid i \in \{1, 2, \ldots, n\}\},$$

where each D_i denotes a local decision table provided by the i-th unit. U_i represents the universe of discourse for the i-th table, consisting of a finite set of objects (instances or records) observed locally. A_i is the set of conditional attributes associated with D_i, capturing the features within the local table. d denotes the decision attribute, assumed to be shared across all decision tables and representing the classification or target variable common to the problem domain, and the global attribute set is given by $A = \bigcup_{i=1}^{n} A_i$, denoting the union of all attributes across the dispersed environment.

2.2 Dispersed Systems Assumptions

The following assumptions characterize the nature and constraints of dispersed data.
To begin, the structure of local decision tables is assumed to be fixed and cannot be externally influenced or modified. Also, while all local tables are related to the same problem domain and thus share an identical set of decision classes, each table also contains attributes and objects unique to it, with the possibility of partial overlap across tables. Furthermore, there is no universal identifier to consistently link objects across different local datasets. Finally, raw local data cannot be directly transferred or concatenated onto a central server, necessitating alternative strategies for effectively making use of dispersed data.

2.3 Methods and Applications

In modern data-driven domains such healthcare, finance and Internet of Things (IoT) applications, data is frequently distributed across multiple independent repositories due to institutional boundaries, privacy constraints, and infrastructural limitations – dispersed data systems. This fragmentation has led to the emergence of learning paradigms aimed at extracting knowledge from decentralized and often heterogeneous data sources without necessitating the centralization of raw data. In the following, an overview of existing approaches designed to address dispersed data systems is presented, emphasizing their methodological foundations, advantages, and limitations.
The two main approaches that have be used for dispersed data in literature are federated learning and ensemble of classifiers.

2.3.1 Federated Learning

Federated Learning (FL), introduced by McMahan et al. [36] has emerged as a prominent decentralized learning paradigm that enables model training across multiple clients without requiring direct access to their raw data. Here, models are trained locally and only model updates (such as gradients or parameters) are transmitted to a central aggregator, which synthesizes a global model. This setup inherently supports privacy preservation and addresses regulatory concerns, especially under data governance frameworks such as GDPR and HIPAA.
However, FL faces several challenges, chief among these is statistical heterogeneity, wherein local datasets follow non-independent and identically distributed (non-IID) distributions [53]. This often results in poor convergence and reduced global model performance. To address these caveats, Li et al. [27] proposed FedProx, which added a proximal term to the local loss function to penalize large deviations from the global model, and in doing so, stabilized convergence under heterogeneous distributions. This approach was proven to demonstrate substantial accuracy improvement over the FedAvg method proposed in [36], however, its effectiveness is highly dependent on the choice of the proximal term. Also, the added proximal term means local devices must compute additional regularization cost during training, which invariably induces extra computational cost. Briggs et al. [5] employed a clustered FL (CFL) approach by adding a hierarchical clustering step into the FL loop. Here, clients first train locally, after which the server clusters clients by the similarity of their model updates to the current global model, and then specialized models are trained per cluster in parallel, yielding faster convergence and a broader client-level accuracy. However, the method's effectiveness is sensitive to the clustering distance metric chosen. Along with this, the method assumes access to full, un-noised client updates – this assumption could degrade clustering in settings that allow differential privacy noise. Another challenge of non-IID data is covariate shift or feature shift,

where distribution of input features varies across clients while the distribution of labels remains relatively stable. Such situations can be found in medical imaging where different hospitals may employ different imaging sensors resulting in feature representations that are not directly comparable. Similarly, in autonomous driving systems, variations in operating environments, such as highways versus urban streets, lead to diverse scene distributions that pose difficulties for robust model generalization. Li et al. [28] proposed FedBN, which addresses this problem by keeping batch normalization local to each client and only aggregating the remaining global model parameters, thereby, preserving client-specific feature statistics which eliminates feature shift effects. However, FedBN only addresses covariate shifts and thus, does not scale or cover all problems associated with non-IID data distributions. Reddi et al. [44] also introduced the Adaptive Federated Optimization (FedOpt) – a general optimization framework for incorporating adaptivity in FL. Here, the authors recast FL as using a client optimizer and a server optimizer, where each client conducts multiple local training epochs using a client-side optimizer to minimize the loss on its private dataset, after which the server refines the global model by employing a gradient-based server optimizer on the aggregated updates from all participating clients. Within this framework, they proposed adaptive server-side methods – FedAdagrad, FedAdam, and FedYogi – that maintain first and second-moment statistics of client updates on the server so clients stay lightweight, making it particulary useful for resource constrained devices. Their theoretical analysis provide convergence guarantees under nonconvex settings, and empirical evaluations across vision and language benchmarks show improved stability and accuracy as compared to other methds like FedAvg [36], however the effectiveness of FedOpt relies heavily on careful hyperparameter tuning, particularly server and client learning rates, which is impractical in dynamic real-world deployments. Also, server-side adaptivity introduces additional state and memory overhead as the server must maintain optimizer statistics for each parameter, unlike FedAvg's stateless aggregation. Dyczkowski et. al [15] proposed an approach that fuses ideas from fuzzy set theory and robust aggregation into the FL paradigm to address application domains such as medical diagnostics, where data are frequently imprecise or incomplete. Here, the authors integrated interval-valued fuzzy representations and similarity-based local update rules into the standard FL cycle. The approach captured measurement imprecision and missingness by encoding the local data at the client level as intervals using fuzzy sets, after which standard gradient aggregation are replaced with similarity-weighted combination rules that give different emphasis to client updates according to their internal uncertainty and mutual agreement. Empirical analysis was performed on the breast-cancer dataset from the UC Irvine repository under multiple missingness and noise scenarios. Results reported in the paper indicate that the uncertainty-aware FL method outperforms baseline FL implementations that treat missing or imprecise values naively, particularly when local datasets are highly incomplete or heterogeneous. Also, by

explicitly modeling local uncertainty and weighting contributions according to similarity, the global model becomes less susceptible to bias introduced by low-quality clients, thus, reducing performance degradation caused by non-IID distributions.

A second major limitation is system heterogeneity, i.e., the variability in computational capabilities, storage, energy resources, and communication bandwidth across participating clients. Unlike controlled centralized environments, federated systems often involve devices ranging from high-performance servers to resource-constrained mobile or edge devices, which can result in stragglers that delay synchronous aggregation rounds [36]. This imbalance not only prolongs training time but can also bias the global model toward clients with greater availability or faster hardware. Kall and Trabelsi [22] proposed an approach (FedAsync) that deviates from the standard synchronous server-driven schemes of FL by updating the global model as soon as any client submits its updates. Their approach integrates personalized local fine-tuning with staleness-aware aggregation, reducing the divergence caused by delayed client updates, and was evaluated in the context of secret detection on GitHub repositories using the Credential Digger tool [49]. The approach showed that personalization is particularly effective under strong heterogeneity, while asynchronous updates improve scalability in dynamic environments. However, the effectiveness of this approach reduces in settings that allow differential privacy noise. Additionally, resource-aware client selection algorithms, such as the approach proposed in [41] dynamically determine which clients to include in each training round based on their computational and communication capabilities. Here, Nishio and Yonetani [41] introduced FedCS, a resource-aware client selection scheme tailored to mobile edge settings. The key innovation is a lightweight, time-constrained selection protocol in which the server first queries candidate clients for their current resource states (e.g., CPU, upload bandwidth) and then selects a subset of clients that can complete local training and return updates within a target round time. By optimizing participant selection under a time budget, FedCS substantially reduces per-round latency and increases the number of effective contributors, making federated rounds more practical and computationally efficient. Despite these strengths, FedCS has several important limitations. First, it relies on timely resource reporting from clients – stale reports may lead to failed rounds or biased selection toward well-resourced nodes. Also, the approach inherently favors high-capability clients, which can harm fairness and representation and bias the global model toward data on powerful devices, thereby reducing the generalizability of the global model.

Communication efficiency is another critical barrier in FL, particularly when dealing with large models or constrained networks. A variety of techniques have been explored to reduce communication overhead. Particularly, to reduce communication overhead and improve convergence rate, Liu et al. [31] introduced a three-tier FL architecture (Hier-FAVG) – comprising clients, edge servers, and the cloud, enabling partial aggregation at

intermediate edge nodes before global aggregation at the cloud server. This hierarchical setup significantly reduces communication overhead and lowers energy consumption for client devices, due to shorter client-edge communication paths and fewer expensive cloud round trips. Despite these contributions, the approach increased system complexity, i.e. deployment and coordination across multiple hierarchical layers (clients, edge, cloud) demand robust orchestration and fault tolerance. Also, the approach does not explicitly address fairness or client selection among unequal edge-layer constraints. Reisizadeh et al. [45] proposed the Federated Learning with Periodic Averaging and Quantization (FedPAQ) method to address the communication bottleneck in large-scale FL systems. It is a communication-efficient algorithm that combines three mechanisms: periodic averaging, partial device participation, and quantized message passing. Specifically, clients perform multiple local updates before synchronizing with the server, thus, reducing communication rounds, while only a fraction of clients participate in each round, and transmitted updates are quantized to reduce bandwidth consumption. Together, these strategies achieve favorable communication-computation trade-offs, with the authors providing theoretical convergence guarantees, achieving $O(1/T)$ for strongly convex loss functions and $O(1/\sqrt{T})$ for non-convex ones. Further, experimental evaluations on standard benchmarks such as MNIST [12] and CIFAR-10 [25] further demonstrate the efficiency gains of FedPAQ over baseline methods. However, the method is highly sensitive to hyperparameter choices, such as the number of local steps and quantization levels, which must be carefully tuned for different data and system settings. Morever, reduced communication leads to slower empirical convergence since FedPAQ may require more iterations to reach target accuracy especially under high-quantization settings. Caldas et al. [6] also addresses communication bottlenecks by introducing two complementary strategies: lossy compression of the global model during server-to-client transmission, thus reducing download size significantly, and a novel technique called Federated Dropout, which enables clients to train on smaller subsets of the global model to reduce local computation while still contributing to the global model. This combination yielded up to 14× reduction in server-to-client communication, 28× reduction in upload communication, and 1.7× decrease in local computation without degradation in model performance. However, the strategy also presents notable limitations. In particular, the use of lossy compression may impact model fidelity in more complex tasks beyond typical benchmarks, and Federated Dropout introduces challenges in maintaining model consistency, especially when clients train only partial models. Additionally, the study primarily evaluates resource savings in relatively homogeneous settings and does not fully address how these optimizations interact with other federated learning challenges, such as non-IID data distributions.

Lastly, privacy and security concerns persist in FL despite the lack of raw data sharing. Studies have shown that model updates can inadvertently leak private information through gradient inversion attacks [54]. To mitigate this, Bonawitz et al. [4] introduced

a practical secure aggregation protocol designed to enhance privacy in FL by enabling a central server to compute the sum of individual client updates without learning any individual contribution (even in the presence of client dropouts). This protocol is both communication-efficient and fault-tolerant, tolerating up to one-third of clients dropping out while still preserving correctness and privacy. The authors provided provable security guarantees under both honest-but-curious and malicious adversarial models, and empirically demonstrate its efficiency: for extremely high-dimensional vectors such, 2^{24} dimensions, communication overhead expands only by 1.98× in worst-case scenarios, which remains practical even at scale. However, while the approach guarantees privacy of updates, the protocol does not prevent leakage from the aggregated global model, leaving it vulnerable to inference attacks post-aggregation. Also, although demonstrated to be efficient, the protocol may still be resource-intensive for very large-scale deployments or extremely constrained edge devices, potentially limiting its applicability in some real-world mobile scenarios. Geyer et al. [19] introduced a federated learning protocol that implements client-level differential privacy (DP), which protects the entire participation status and dataset contributions of clients. The core mechanism incorporates randomized client sub-sampling and the addition of Gaussian noise to update aggregates, managed via a privacy curator that dynamically tracks cumulative privacy loss. This design ensures that the model conceals whether any particular client participated in training. The authors provided empirical studies that showed that given a sufficiently large number of participating clients, their proposed method can maintain client-level differential privacy at a minor cost in model performance, making it viable for deployment in privacy-critical cross-silo scenarios. However, since the method relies on a large number of participants, it may not be feasible in smaller federated settings with few clients, such as medical institutions.

2.3.2 Distributed Learning

Distributed learning, on the other hand, assumes full access to data at a central location but subdivides it into chunks for parallel processing or model enhancement. Distributed Learning (DL) refers to frameworks where training is parallelized across multiple machines or nodes, each handling a subset of the data or model. Unlike federated learning, distributed learning assumes full access to data at a central location (subdivided it into chunks across nodes) and focuses on improving computational efficiency and model enhancement.

Early work on distributed learning focused on parallelizing training across multiple machines to accelerate model convergence, especially for large-scale deep learning models. Dean et al. [10] presented DistBelief, a pioneering software framework for distributed deep learning that enables training neural networks with billions of parameters across tens of

thousands of CPU cores. The framework introduced two key algorithmic innovations: Downpour SGD, an asynchronous stochastic gradient descent allowing highly scalable model replication and gradient updates without global synchronization, and Sandblaster, a system supporting distributed batch optimization methods like L-BFGS [30]. Demonstrating remarkable scalability, the authors trained models 30× larger than previously reported in the literature and achieved state-of-the-art performance on ImageNet [11]. This work represents one of the earliest demonstrations of practical, large-scale distributed deep learning and laid the groundwork for modern parallel training methods.

Despite its scalability benefits, distributed learning systems face several notable challenges. A major challenge in DL is communication overhead. Frequent communication between nodes, especially in synchronous setups, can result in network congestion and delay. This becomes a bottleneck in bandwidth-constrained or latency-sensitive environments. To mitigate this, several communication-efficient strategies have been proposed – Seide et al. [50] introduced 1-bit stochastic gradient descent (1-bit SGD), a communication-efficient training strategy that dramatically reduces gradient exchange overhead in distributed deep learning. The key innovation lies in quantizing gradients to a single bit per value, while employing an error-feedback mechanism to accumulate and correct quantization errors across iterations. This approach reduces communication costs by up to 32× without significant accuracy loss. Building on this, the authors designed a scalable data-parallel SGD framework that combines AdaGrad, dynamic batch scaling, double buffering, and model parallelism, enabling efficient training of large-scale speech recognition models with hundreds of millions of parameters. Their system achieved speedups of up to 6× compared to single-GPU [48] baselines, completing training runs involving thousands of hours of speech data in a fraction of the usual time. Despite its effectiveness, the implementation also introduces engineering complexity, as it requires the integration of multiple optimization and communication strategies, which may hinder broad adoption. Also, the approach depend heavily on high-performance GPU [48] clusters and large batch sizes, limiting the method's applicability in resource-constrained environments. Similarly, Lin et al. [29] proposed Deep Gradient Compression (DGC), which addresses the critical bottleneck of communication overhead in distributed training by recognizing that up to 99.9% of gradient data exchanged during SGD is redundant. By retaining only the top-magnitude gradients, coupled with mechanisms such as momentum correction, local gradient clipping, momentum masking, and warm-up training, DGC achieves dramatic communication savings – up to a 600× reduction in gradient size, reducing ResNet-50 [24] updates from 97 MB to 0.35 MB – without any loss in model accuracy. Empirical evaluations on benchmarks including CIFAR-10 [25], ImageNet [11], Penn Treebank [33], and LibriSpeech [42] confirm that DGC preserves convergence behavior while enabling distributed training on commodity 1 Gbps Ethernet and mobile devices. However, the method's reliance on gradient sparsification and its corrective

mechanisms introduces added complexity that may complicate implementation and tuning, particularly under evolving training dynamics. The sparsity assumption may not hold uniformly across all architectures or tasks as certain models or optimization regimes could be more sensitive to aggressive compression.

Another limitation of DL is the node failures in DL. Fault tolerance is a critical component in distributed learning systems due to the high likelihood of node failures, network disruptions, and hardware inconsistencies. These failures can result in partial model updates, lost gradients, or system downtime, ultimately affecting the reliability and convergence of learning algorithms. To mitigate this, Cipar et al. [8] introduced Stale Synchronous Parallel (SSP), an enhanced model of iterative parallel computation designed to address the straggler problem often encountered in Bulk Synchronous Parallel (BSP) systems [18]. SSP generalizes BSP by permitting bounded staleness in updates, in that slower worker threads are allowed to lag behind others by a fixed number of iterations, without forcing global synchronization at each step, hence, upon failure, worker nodes can resume from the last checkpoint with minimal disruption. This flexibility enables systems to continue efficient progress even with variability in computation time across worker nodes. Experiments with a prototype system named "LazyTables"demonstrate that SSP significantly reduces delays caused by stragglers in tasks such as sparse regression and low-rank matrix factorization, demonstrating enhanced system efficiency.Despite these advantages, SSP requires careful tuning of the staleness threshold – a trade-off between reducing straggler impact and avoiding excessive divergence in worker progress. Additionally, SSP introduces additional overhead in maintaining worker lag tracking and consistency, increasing system complexity compared to BSP [18].

From a data redundancy perspective, Roy et al. [46] introduced a lightweight partial-replication-based recovery strategy method in decentralized social networks when an individual user node fails – instead of fully replicating all node data, only selected portions are redundantly maintained among peers, facilitating swift node recovery without requiring full restarts while minimizing memory overhead and system downtime especially in systems where users operate autonomously, and full data duplication would be expensive. Despite these advantages, this method exhibit some limitations. Specifically, by concentrating on single-node failures, it does not extend to scenarios involving simultaneous multiple-node outages, which could significantly challenge network stability. Also, partial replication, while storage-efficient, may compromise data completeness or consistency if not carefully managed, especially under conflicting updates.

While existing approaches in FL and DL have demonstrated effectiveness in various settings, a significant limitation persists in scenarios involving fully heterogeneous datasets – where neither the object sets nor the attribute spaces overlap across local nodes. Most FL algorithms, such as FedAvg [36], rely on assumptions of partial data alignment or necessitate complex synchronization mechanisms, which may inadvertently compromise data

privacy [37]. In contrast, traditional DL paradigms typically presume centralized access to the entire dataset, which is then partitioned and distributed across computational nodes – an assumption that is often impractical in privacy-sensitive or siloed environments. This dissertation aims to fill this gap by proposing a range of neural network-based methods and data augmentation techniques that assumes neither feature-space homogeneity nor centralized access.

Chapter 3

Proposed Approaches

This section focuses on the set of methods proposed to overcome the challenges of dispersed data that FL and DL fail to address.

3.1 Fusion of Local Predictions Using Neural Networks

In the context of learning from dispersed data environments, where datasets are fragmented across independent sources, the exploration on the use of neural fusion strategies to aggregate local model predictions was carried out. These efforts progressively advanced from foundational fusion architectures to more domain-specific and robust designs, with growing attention to architectural efficiency, data heterogeneity, and domain adaptation. The first approach in this line of research introduced a fusion framework leveraging the Multilayer Perceptron (MLP) neural network [43] to combine predictions from independent k-NN classifiers trained on non-overlapping datasets in the paper [**P1**]. The method combined local k-NN classifiers with a neural network-based fusion mechanism, where the similarity between p-dimensional objects x and y of mixed attributes $\{a_i, \cdots, a_p\}$ is measured using the Gower similarity measure:

$$S(x, y) = \frac{\sum_{i=1}^{p} s_i(x, y)}{\sum_{i=1}^{p} \delta_i}$$

where $s_i(x, y)$ denotes the contributions by the i-th attribute defined as:

- For quantitative attributes:

$$s_i(x, y) = 1 - \frac{|a_i(x) - a_i(y)|}{\text{range of values for attribute } i}$$

- For nominal attributes

$$s_i(x, y) = \begin{cases} 1 & \text{if } a_i(x) = a_i(y), \\ 0 & \text{otherwise.} \end{cases}$$

with

$$\delta_i = \begin{cases} 1 & \text{if } s_i(x, y) \text{ is defined} \\ 0 & \text{otherwise.} \end{cases}$$

The Gower similarity measure is flexible enough to accommodate mixed attribute types by assigning binary matches for nominal values and scaled distance reductions for numerical ones. Each k-NN produces a probability distribution over decision classes as shown in Algorithm 1, and these outputs are concatenated and passed as input features to a MLP network, enabling the learning of high-level inter-dependencies and non-linear mappings among the distributed predictions. Figure 3.1 presents the stages in the approach described above. Five different dispersed versions with 3, 5, 7, 9 and 11 local tables

Algorithm 1 Pseudo-code of algorithm generating predictions based on local tables

Input: A set of local decision tables $D_{ag} = (U_{ag}, A_{ag}, d), ag \in Ag$; test set - a decision table $D_{test} = (U_{test}, A_{test}, d_{test})$, $A_{test} = \bigcup_{ag \in Ag} A_{ag}$

Output: A set of vectors over decision classes $\mu_{ag}(x), ag \in Ag, x \in U_{test}$.

foreach $x \in U_{test}$

foreach $ag \in Ag$

Calculate the value of the Gower similarity measure for each object in the set U_{ag} and the test object x (using only the attributes from the set A_{ag}).

For each j-th decision class, find the k nearest neighbors objects to x from j-th decision class of the decision table D_{ag}.

$\mu_{ag,j}(x)$ is equal to the mean of similarity of the k nearest neighbors objects from the j-th decision class of the decision table D_{ag}.

end foreach

end foreach

were prepared from benchmark datasets such as Lymphography [55], Vehicle Silhouettes [40], and Soybean (Large) [38] and experimental results showed robust performance, with accuracies reaching 90.7% on Soybean and remaining stable across different levels of dispersion – 0.913 to 0.902 when increasing from 3 to 11 tables). The method consistently outperformed classical fusion techniques such as majority voting [16], Borda count [47], and decision templates [26] – generating unambiguous decisions over test objects. The innovation of the approach lies in using a neural architecture to unify dispersed probability outputs while retaining robustness to partitioning and heterogeneity. However,

the approach introduces interpretability challenges due to the "black-box"nature of the MLP. Also, its effectiveness is strongly dependent on the calibration quality of local k-NN predictions, making it vulnerable to the propagation of local model errors. Moreover, due to the inherent complexity of MLPs, the approach risked overfitting, particularly in scenarios with limited fusion training data.

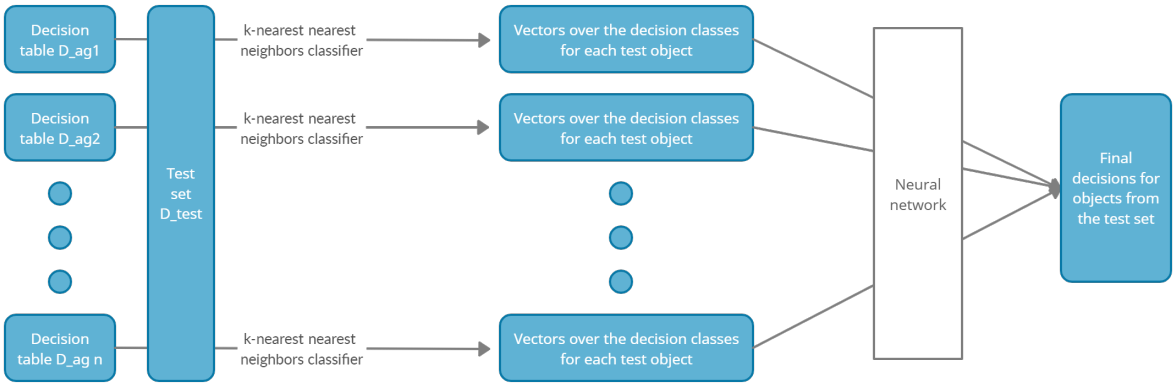


Figure 3.1: Stages in the dispersed classification method

In the paper [**P2**], research was carried out to test the robustness of the approach proposed in [**P1**] by empirically investigating how data characteristics and noise affect classification quality. The study systematically generated 270 synthetic datasets that vary along five factors: number of conditional attributes, number of objects, number of decision classes, degree of dispersion (number of local tables), and noise intensity. Specifically, synthetic datasets were generated using the RandomRBF function in Weka [17]. This generator assigned random centers, weights, attribute means, and standard deviations to decision classes, from which new objects are sampled by selecting a center according to the weights initially assigned, perturbing attribute values, and scaling them by a Gaussian-distributed length. Using this procedure, 18 datasets were created with varying numbers of conditional attributes, decision classes, objects and centroids, with characteristics summarized in Table 3.1. Each dataset was stratified into training (70%) and testing (30%) subsets. To evaluate the influence of noise intensity of classification quality, Gaussian noise with mean 0 and standard deviations $\{0.01, 0.1, 0.2\}$ was applied to the training sets, yielding 54 noisy datasets. To examine the impact of dispersion, each of these datasets was partitioned into $3, 5, 7, 9$, and 11 local decision tables, where each table contained both unique and overlapping conditional attribute, ranging from 3 to 35 attributes. This procedure resulted in 270 dispersed datasets, representing combinations of noise intensity levels and dispersion degrees. Experimental analysis highlighted several key findings. Among the dataset characteristics, the number of conditional attributes had the strongest influence on classification quality, with higher dimensionality generally

Table 3.1: Synthetic Data set characteristics

Data Set	No. of Objects	No. of Conditional Attributes	No. of Decision Classes	No. of Centroids
1	650	30	10	100
2	650	50	10	100
3	650	70	10	100
4	1300	30	10	100
5	1300	50	10	100
6	1300	70	10	100
7	650	30	5	50
8	650	50	5	50
9	650	70	5	50
10	1300	30	5	50
11	1300	50	5	50
12	1300	70	5	50
13	650	30	5	100
14	650	50	5	100
15	650	70	5	100
16	1300	30	5	100
17	1300	50	5	100
18	1300	70	5	100

improving performance. The number of decision classes was the second most important factor, as fewer classes yield easier and more accurate classification, while the number of training objects plays a comparatively minor role, though larger datasets slightly improve results. The degree of dispersion also affected performance: high dispersion such as 11 local tables led to a substantial decline in classification accuracy, whereas low to moderate dispersion (Tables 3–7) has only a marginal effect. Noise similarly degraded performance, although the method shows partial robustness – for Gaussian noise with mean 0 and standard deviation ≤ 0.1, classification accuracy remains relatively stable, but with $std = 0.2$, performance deteriorates sharply, indicating limited tolerance to high noise levels. These findings offered empirical validation of the method's robustness, while also identifying key failure modes under extreme data heterogeneity or imbalance.

To address the challenges associated with the approach in **[P1]**, a subsequent paper **[P3]** proposed an alternative fusion strategy based on Radial Basis Function (RBF) neural networks [52]. RBFs offer a distance-sensitive mechanism for prediction fusion, treating input probability vectors as spatial points and computing their similarity to learned centers using Gaussian functions. Similar to the approach in **[P1]**, at the local level, similarity between training objects were computed using the Gower similarity measure, however, here, the probability vectors over decision classes are concatenated and passed to a RBF network as its input. The normalized Gaussian function was used as the radial kernel, given as:

$$\Phi_i(y) = \exp\left[- \frac{\| y - c_i \|}{2\sigma_i^2} \right],$$

where y is the input vector (prediction vectors), c_i is the center of the i-th neuron approximated using Lloyd's k-means algorithm [32], σ_i is the Gaussian width of i-th neuron estimated as $\sigma_i = \dfrac{\rho_{\max}}{2n_H}$ where $\rho_{\max}$ is the maximum distance between the chosen centers and n_H is the total number of centers – equivalently, the number of neurons in the hidden layer. This architecture provided a more localized and interpretable fusion process, enabling better handling of subtle prediction variations and conflicting local outputs. Additionally, the simplicity of the RBF structure supported faster convergence and reduced

overfitting risks due to the fewer trainable parameters. Experimental analysis was carried out on four datasets – Lymphography [55], Vehicle Silhouettes [40] and two synthetic data sets created using the RandomRBF method from Weka [17], data characteristics given in Table 3.2. Each of the datasets was partitioned into $3, 5, 7, 9$, and 11 local decision

Table 3.2: Characteristics of Data sets

Data Set	# Training Set	# Test Set	# Attributes	# Decision Classes
Vehicle Silhouettes	592	254	18	4
Lymphography	104	44	18	4
Artificial Data 1	500	150	30	10
Artificial Data 2	500	150	50	10

tables and three different values of the k parameter were tested, i.e $k \in \{1, 5, 10\}$. A comparative evaluation of MLP and RBF-based methods were carried out and the results indicated that the RBF-based aggregation consistently outperformed the previously proposed MLP fusion approach in [**P1**] across multiple datasets and dispersion settings. For instance, on the Vehicle dataset, the RBF achieved a minimal error of 0.242 (5-NN, 5 local tables), compared to 0.329 for MLP under the same configuration. Similar improvements were observed on Artificial Data 1, where the RBF reached 0.038 versus the MLP's 0.087, and on Artificial Data 2, where the RBF achieved 0.016 against the MLP's 0.024. A Wilcoxon signed-rank test over 40 paired observations confirmed the statistical significance of these improvements ($p = 0.0003$), highlighting the robustness of the RBF model in dispersed data environments. While both methods generated optimal classification quality, the RBF-based approach provided lower error rates and reduced structural complexity, however, despite these advantages, the RBF-based method introduced its own set of challenges, particularly in high-dimensional settings. Its performance was sensitive to center selection and kernel width configuration, and generalizing effectively across different datasets required careful tuning.

Recognizing the limitations of heuristically chosen RBF centers – especially in high-dimensional spaces like images – the study [**P4**] extended the RBF approach in [**P3**] to dispersed image classification by introducing a supervised center training phase for the RBF network. Unlike the earlier approaches in [**P3**] that relied on heuristic or random center initialization, this method learned optimal RBF centers from data by back-propagating the error up to the centers during training, leading to improved alignment with class structures and more accurate predictions. This adaptation significantly enhanced the model's ability to manage high-dimensional image features and class boundaries across disjoint visual datasets. Particularly, the experimental analysis was carried out on benchmark datasets including Vehicle Silhouettes [40], Statlog (Landsat Satellite) [51] and Dry Bean [3], each dispersed into multiple local decision tables – $\{3, 5, 7, 9, 11\}$,

to simulate heterogeneous environments. The evaluation considered classification accuracy across varying numbers of local tables, levels of dispersion, and different network configurations, with hidden layer sizes scaled relative to input dimensionality. Comparative baselines included the vanilla RBF model from **[P3]**, relying solely on k-means initialization of centers, and an ensemble of classifiers combining k-NN, Decision Tree, and Naive Bayes. The results demonstrated that the proposed RBF with centers training consistently achieved the highest classification accuracies across all datasets and dispersion settings, such as 0.769 on Vehicle (3 local tables), 0.897 on Satellite, and 0.920 on Dry Bean, outperforming both vanilla RBF (0.656, 0.853, 0.900 respectively) and the ensemble (0.657, 0.885, 0.906 respectively). Also, these improved accuracies were obtained with reduced network complexity, as the proposed model required fewer hidden neurons to achieve optimal performance. Statistical tests, including the Friedman and Wilcoxon analyses, confirmed the superiority of the proposed approach ($p < 0.05$) over both baselines, while no significant difference was observed between the vanilla RBF and the ensemble. While effective, the added center training phase introduced computational overhead, increasing the optimization complexity of the overall pipeline. Nevertheless, the work demonstrated the adaptability of neural fusion to visual domains and underscored the benefits of learning the fusion mechanism itself rather than fixing it apriori.

The last approach in this line of research addressed a fundamental structural issue in dispersed learning, i.e, object distribution heterogeneity across local datasets. The papers **[P7]** and **[P10]** investigated how varying degrees of object overlap and feature diversity among datasets influence the classification quality. This study examined the partitioning of objects across local tables in order to progressively reduce the proportion of common objects. Initially, all training data reside in a single table, which is then divided into multiple local tables with disjoint attribute sets, while allowing a fraction of attributes to overlap across tables. The procedure, described in Algorithm 2 first allocates an equal share of unique objects to each local table in a stratified manner. Subsequently, a proportion p of objects is drawn from each table and redistributed equally among the remaining tables, thereby creating a fully heterogeneous dispersed system. For instance, given a dataset M of n objects partitioned into three tables T_1, T_2, T_3, each T_i initially receives $\lfloor n/3 \rfloor$ objects, after which a fraction p of its objects is shared equally with the other two. This way, we would have objects common between tables (T_1, T_2), (T_1, T_3), and (T_2, T_3). Similar to the experimental setup in **[P4]**, three datasets from the UCI repository [14] (Vehicle Silhouettes [40], Statlog (Landsat Satellite) [51] and Dry Bean [3]) were dispersed into $3, 5, 7, 9$, and 11 local tables, with common objects reduced from 45% to 30% and 15%. Performance was evaluated using accuracy, F-measure, precision, recall, and balanced accuracy, across both imbalanced and SMOTE-balanced [7] variants of the datasets. Experimental results showed that object diversity had minimal effect on classification quality. Statistical analysis using the Friedman test revealed no significant

differences in accuracy or F-measure across 45%, 30%, and 15% common objects, with only balanced accuracy showing significance ($\chi^2(90, 2) = 7.058$, $p = 0.03$). For instance, in the Satellite dataset with $k = 1$, accuracy remained consistently high: 0.904 (45% common), 0.899 (30%), and 0.903 (15%). Similarly, Dry Bean results were remarkably stable, achieving accuracy of 0.919-0.921 across all configurations. The Vehicle dataset showed slightly more variability but still maintained competitive performance with F-measure ranging from 0.748 to 0.769 at $k = 1$. Interestingly, imbalanced data often yielded slightly better results than balanced data with Wilcoxon tests confirming significant differences ($p = 0.0001$) in all metrics between balanced and imbalanced conditions, demonstrating the robustness of the method to object diversity in local tables as well as data imbalance.

Algorithm 2 Algorithm of division objects between local tables

Input: A set of objects U with cardinality N,
p - percentage of diversification, percentage of common objects,
k - number of tables.
Output: Universe of local tables, $U_i, i \in \{1, \ldots, k\}$.

numOfLocalObj = $\left\lfloor \frac{N}{k} \right\rfloor$;
remainingObj = $N - (k \cdot \text{numOfLocalObj})$;
We divide objects from U in a stratified and disjoint manner into k subsets U_i each containing numOfLocalObj;
Objects from the set U that are not included in any local set U_i are added to all local sets U_i, their number is equal to remainingObj;
for each $i = 1 \ldots k$ do
$\rightarrow$ Choose p percent of the object from the set U_i;
$\rightarrow$ Save in S_i;
for each $i = 1 \ldots k$ do
$\rightarrow$ $U_i = U_i \cup \bigcup_{j \in \{1,\ldots,k\}, j \neq i} \left\lfloor \frac{S_j}{(k-1)} \right\rfloor$
Return $U_i, i \in \{1, \ldots, k\}$

The results presented in this section allow the first research question to be answered affirmatively. Neural networks prove to be effective fusion mechanisms, with the MLP-based model achieving high and stable classification accuracy across a range of datasets and dispersion levels. Also, the comparative analysis indicate that the RBF network offers a superior alternative. The RBF model consistently produced lower classification errors, with statistically significant improvements over the MLP, and benefited from reduced architectural complexity and a lower risk of overfitting. Thus, while both architectures successfully fuse predictions from dispersed sources, the RBF network emerges as the more accurate and robust choice.

3.2 Aggregation of Local Neural Networks

To overcome the over-reliance on k-NN prediction vectors in the previous section, this line of research introduced a novel strategy for building a single global MLP model on dispersed tabular data, where local datasets differ not only in the set of attributes but also in the set of observed objects in the papers [**P5**], [**P6**] and [**P9**]. The idea here is to build a model for each decision table and aggregate their trained parameters to create a global model, however, since the structure of decision tables are heterogeneous, the principal challenge here lies in standardizing the input architecture across all local models. This was addressed through the use of artificially generated objects, which filled in missing attribute values via imputation, as described in Algorithm 3, with Figure 3.2 showing the stages of building the global model. The computational complexity of Algorithm 3 scales linearly with the number of objects in a local table D_j, the parameter k, the number of conditional attributes $\text{card}\{A\}$, and the number of local tables n. Formally, the loop complexity is

$$O(\text{card}\{U_j\} \cdot k \cdot \text{card}\{A \setminus A_j\} \cdot \text{card}\{D_i, i = 1, \ldots, n\})$$

In the worst case, where D_j contains only one conditional attribute, and all remaining attributes must be computed from the other $n - 1$ tables, the complexity reduces to

$$O(\text{card}\{U_j\} \cdot k \cdot (\text{card}\{A\} - 1) \cdot (n - 1)),$$

demonstrating the algorithm's scalability and suitability for large dispersed datasets. These synthesis allowed consistent input-layer configurations across all decision tables, enabling global model aggregation without requiring the exchange of raw data while still supporting joint learning. Experiments used three benchmark datasets: Vehicle Silhouettes [40], Statlog (Landsat Satellite) [51] and Dry Bean [3], each dispersed into multiple local decision tables – $\{3, 5, 7, 9, 11\}$. The study systematically varied the number of artificially generated objects per original instance ranging from 1 to 5 across both imbalanced and balanced variants of the datasets with the hidden layer sizes parameterized based on the input layer from $0.25 - 5\times$ input layer size. Experimental analysis showed that the optimal number of artificial objects depends on dataset size: for smaller datasets such as Vehicle, generating more artificial objects improved accuracy up to 0.773 with five artificial objects, whereas for larger datasets like Dry Bean, only two artificial objects were sufficient to reach peak performance of 0.917. Dispersion effects were dataset-dependent; for Landsat and Dry Bean accuracies varied only in the third decimal place across $3 - 11$ local tables, but for Vehicle dispersion influenced results more substantially, with statistically significant differences confirmed by Friedman tests with $\chi^2 = 26.608, p = 0.00003$. Finally, larger hidden layers up to $5\times$ the input size consistently improved accuracy, un-

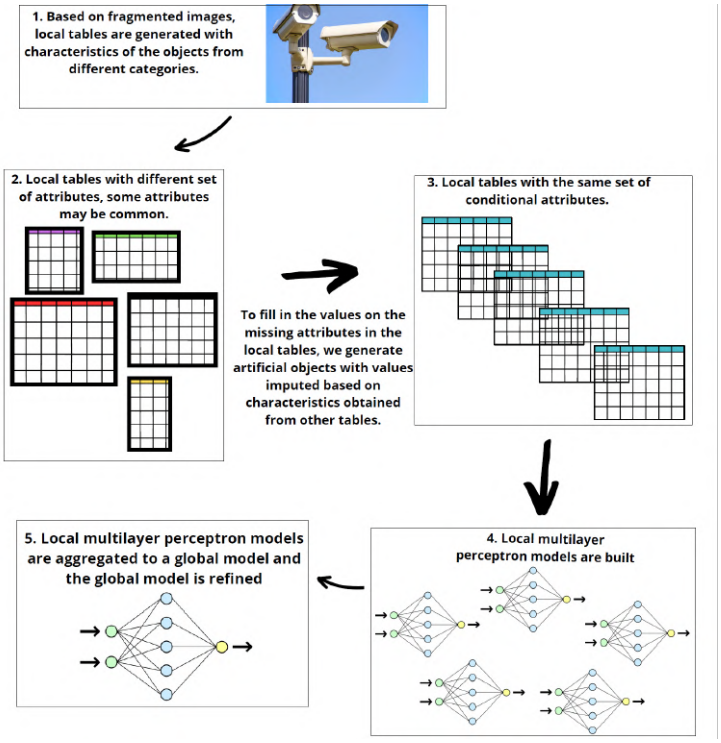


Figure 3.2: Stages of building a global model.

derscoring that model capacity is critical when abundant data are available. However, the method also presented key limitations: its success depended heavily on the generated artificial objects. Poorly imputed values could introduce biases, especially if simplistic assumptions governed the generation process. Additionally, the approach required careful calibration of both the number of artificial instances and the hidden-layer size, making it sensitive to dataset complexity. These findings highlighted both the effectiveness and the dataset sensitivity of artificial-object augmentation as a strategy for dispersed learning.

Algorithm 3 Pseudo-code of algorithm generating objects from one local table used for building MLP models

Input: One local decision table $D_j = (U_j, A_j, d)$ for which we determine the training set for the MLP network; measures minimum, maximum, median and average $MIN^b_{i,v}$, $MAX^b_{i,v}$, $MED^b_{i,v}$ and $AVG^b_{i,v}$ computed for each decision value $v \in V^d$ and attribute $b \in A_i$ based on the values stored in the table D_i for each $i \in \{1, \ldots, n\}$; $A = \bigcup_{i=1}^{n} A_i$ a set of conditional attributes from all local tables; k parameter value that determines how many objects are generated based on one object from table D_i.
Output: A data set used to train the MLP neural network, $\overline{D_j} = (\overline{U_j}, A, d)$.

foreach $x \in U_j$

for $m = 1$ to k do

create an object $\bar{x}$ from $\overline{D_j}$ by assigning values on the set A_j the same as the object x has

foreach attribute in the set $b \in A \setminus A_j$

choose a pair (choice1, choice2) from the set
{MIN, MAX, AVG, MED} $\times$ {MIN, MAX, AVG, MED}
$b(\bar{x}) = \text{choice2}_{i:b \in A_i}\ (\text{choice1}^b_{i,d(x)})$

end foreach

end foreach

end foreach

In summary, the experimental results discussed in this section positively answers the second research question. Indeed, the strategy of imputing missing attributes successfully harmonized heterogeneous local datasets, enabling the construction of a unified global neural network through the aggregation of locally trained weights. Experimental analysis showed that this approach improved classification accuracy relative to models trained exclusively on incomplete or non-aligned local data. By reconstructing a common feature space, the method reduced information loss caused by attribute heterogeneity. Additionally, the aggregated network exhibited competitive performance even when the proportion of artificially generated objects was substantial, demonstrating the robustness of the approach.

3.3 Feature Extraction Models

To overcome the computational overhead incurred from generating artificial objects for imputation described in the previous section, this line of research proposed a generic hierarchical model by introducing a two-level neural architecture framework that balances data privacy with predictive accuracy in the papers [**P8**] and [**P11**]. The process consists

of two main levels:

Level 1: Local Table Transformation

A central innovation of the proposed framework lies in its treatment of heterogeneous data through local table transformation. Each decision table often differs not only in size but also in attribute composition. To enable interoperability, each local table is independently projected into a unified k-dimensional feature space without any exchange of raw attributes or transformation parameters. Any data transformation method can be used in this step, however in this research, four complementary dimensionality reduction techniques were used, namely, Principal Component Analysis (PCA) [34], Singular Value Decomposition (SVD) [23], and Uniform Manifold Approximation and Projection (UMAP) [35]. PCA and SVD provided linear projections that captured global variance and latent structure, while UMAP offered a nonlinear embedding that preserved both local and global topology. Only PCA was used in the paper **[P8]** as the transformation map, which was later extended in **[P11]** to include SVD and UMAP. By concatenating these transformed representations,

$$\mathcal{C}_i^k = [x_{PCA}^k, x_{SVD}^k, x_{UMAP}^k],$$

the framework recovers complementary structural information while maintaining data confidentiality. Importantly, transformation is performed entirely locally, ensuring that original features remain secluded from other decision tables This transformation strategy addresses the fundamental challenge of dispersed environments: attribute-space heterogeneity. Unlike federated learning approaches that presuppose feature alignment or rely on costly synchronization, the proposed method achieves compatibility at the representation level, enabling collaboration across structurally incompatible datasets. Also, since the transformation maps the local data into a shared latent space, it enables a common representational basis across heterogeneous sources by semantically aligning disparate local feature spaces without requiring strict feature matching, thus overcoming the need to generate artificial objects from imputation.

Level 2: Global Model Training

Separate models are trained independently in each data center using transformed datasets described in 3.3, ensuring that the structure of local models does not reveal sensitive information about the original data attributes. This hierarchical approach offers an effective compromise between local autonomy and global model coherence. It supports data heterogeneity, limits privacy exposure, and simplifies network design by standardizing input vectors.

Particularly, in **[P11]**, following the transformation sate in 3.3, independent neural network models are trained locally on the unified feature tables $\mathcal{C}_i^k$. The framework is architecture-agnostic, supporting both feedforward and recurrent neural networks. Experiments were carried out on three datasets from the UCI repository [14] (Statlog (Landsat Satellite) [51], Crowdsourced Mapping [21] and Vehicle Silhouettes [9]) each dispersed into multiple local decision tables – $\{3, 5, 7, 9, 11\}$. Three transformation techniques (PCA, SVD, UMAP) and five neural architectures, namely Multilayer Perceptron (MLP) [43], Simple Recurrent Network (SIMPLE) [39], Gated Recurrent Unit (GRU) [13], Long Short-Term Memory (LSTM) [20], and Radial Basis Function Network (RBF) [52] were evaluated. The experimental design varied: the number of hidden layers (1-3), the number of principal components k (ranging from 1 to the feature size of the smallest local table), and the neurons in the hidden layers, scaled relative to the input dimension. For 1 hidden layer, scaling factors were $\{4, 6, 9, 12, 20\}\times$ input size; for 2 layers, $\{(4, 2), (5, 3), (6, 3), (8, 4), (10, 5), (12, 6), (15, 9), (20, 10)\}\times$ input size and for 3 layers, $\{(4, 3, 2), (6, 5, 4), (7, 5, 3), (8, 4, 2), (10, 7, 3), (12, 6, 3), (15, 9, 4), (20, 10, 5)\}\times$ input size. For the MLP models, 1-3 hidden layers were explored, recurrent models (GRU, LSTM, SIMPLE) were restricted to shallow architectures (1-2 layers) to mitigate vanishing gradient and overfitting risks, while RBF networks, by definition, employed a single hidden layer. At inference, each transformed test instance $\mathcal{X}^k = [x_{PCA}^k, x_{SVD}^k, x_{UMAP}^k]$ is fed into the locally trained models. Their predictions are aggregated via soft voting:

$$\hat{y} = \frac{1}{n} \sum_{i=1}^{n} f_i(\mathcal{X}^k),$$

where f_i denotes the prediction function of the i-th local model as described in Figure 3.3.

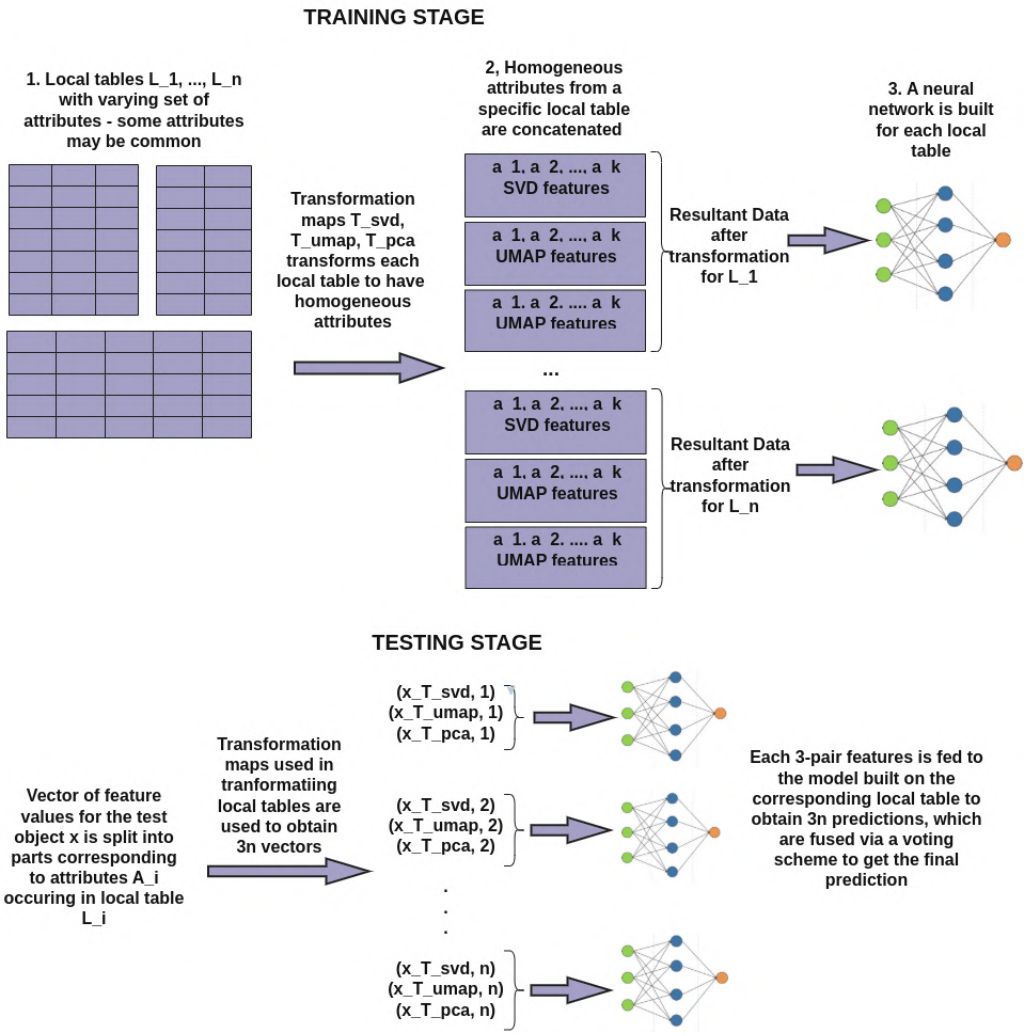


Figure 3.3: Model generation and prediction for test object stages

This averaging strategy avoids parameter synchronization while still leveraging model diversity, yielding robust global predictions. Experimental results showed that the MLP and SIMPLE models achieved the strongest performance, with MLP reaching 0.855 accuracy on Satellite and 0.914 on Anuran, while SIMPLE achieved comparable or better results in several configurations like 0.866 accuracy on Satellite, 0.911 on Anuran. GRU models performed competitively, often ranking just below MLP and SIMPLE, while LSTM achieved slightly weaker results, reflecting higher computational overhead. RBF networks showed stable performance, particularly with larger feature dimensions k. Statistical analysis confirmed significant performance differences among architectures (Friedman test, $\chi^2(4, 76) = 42.70, p = 0.00001$), with MLP, SIMPLE, and GRU consistently outperforming others. Higher values of k significantly improved classification quality (Kruskal–Wallis test, $H(5) = 45.33, p < 0.00001$), validating the benefit of richer feature representations. Overall, the experiments demonstrated that the proposed framework is scalable, resilient to dispersion, and effective at preserving privacy without relying on shared features or

synchronized updates.

The findings in this section confirms a positive answer to the third research question. By transforming each local dataset into a common feature space using feature extraction methods, a global model that achieves strong and reliable classification accuracy was constructed. By standardizing heterogeneous attribute sets without generating artificial objects, the method simplifies the integration of local datasets while preserving the data structure. Experiments showed that models trained on these transformed features, and fused through soft voting, consistently produced satisfactory results, with richer transformation spaces yielding additional gains. Thus, feature-extraction based models offered an effective and precise approach for classification in dispersed data settings.

Chapter 4

Summary

Research on dispersed data environments seeks to develop classification models capable of operating when data are distributed across multiple independent sources, often with heterogeneous structures that cannot be centralized due to data privacy laws such as GDPR [2]. Real-world datasets are rarely homogeneous, characterized by varying feature sets, varying object distributions, and strict privacy constraints. Classical machine learning methods, which typically assume homogeneous and centrally available data, struggle under such conditions. Consequently, designing learning procedures that are robust to structural incompatibility and dispersion is essential for transforming these fragmented datasets into accurate and practical classification systems.

In this thesis, a sequence of complementary methods addressing different facets of dispersed learning was proposed. Each of the developed approaches emerged directly from limitations observed in real systems, where data could not be shared or aligned in a trivial way. The resulting algorithms were examined using both synthetic and real datasets and were shown to behave as intended, delivering improved and consistent classification performance even in highly heterogeneous settings.

The first group of studies focused on the fusion of local predictions. When each independent data source train its own classifier, the challenge lies in generating a coherent global decision without exchanging raw attributes. In [**P1**], a method was introduced in which each local dataset trains a k-NN classifier, and the resulting prediction vectors are subsequently integrated using a neural network. This approach not only mitigated the challenges related to data privacy and structural heterogeneity of data, but with the neural network component effectively capturing intricate patterns, the approach generated unambiguous decisions. This method introduced in [**P1**] was extended in [**P2**], which demonstrated how noise intensities, varying data characteristics, and different levels of dispersion affect the stability of the fusion process, demonstrating the robustness of the approach to noise perturbations and various data dispersion levels. Later, [**P3**] and [**P4**] enhanced this concept by replacing traditional MLP fusion with RBF networks, including a dedicated training strategy for tuning RBF centers. This substantially reduced

architectural complexity, facilitated faster convergence, and mitigated the risk of overfitting commonly associated with MLP-based methods. These contributions collectively provided a set of reliable tools for fusing predictions originating from heterogeneous local datasets, even when the underlying data structures differ substantially.

The next group of publications shifted the focus toward constructing a single global model from dispersed datasets. Instead of relying on prediction fusion, the objective was to standardize the structure of local decision tables so that local neural networks could be aggregated into a global one. A key difficulty addressed here stems from the fact that local tables may contain entirely different sets of conditional attributes. Paper [**P5**] introduced a constructive imputation method based on artificially generated objects that fill missing attributes using statistical characteristics gathered from other local tables. This strategy makes it possible to train structurally aligned neural networks locally and later combine their parameters in a single aggregation step. These ideas were expanded in [**P6**] and [**P9**], where the quantity and diversity of artificial objects were analyzed in depth, and guidelines for designing effective MLP architectures for such global model construction were proposed.

The last group of studies shifted from artificial imputation of local tables to feature-extraction based harmonization. Instead of augmenting datasets with synthetic objects, each local table was transformed into a shared feature space using dimensionality reduction methods. In [**P8**], the use of PCA was proposed to map every dataset into a consistent k-dimensional representation, allowing subsequent fusion of predictions from MLP models through soft voting, while ensuring data privacy. This idea was significantly expanded in [**P11**], where a broader suite of feature-extraction methods including PCA, SVD and UMAP and neural architectures such as MLP, RBF, GRU, LSTM, SIMPLE were systematically evaluated. These studies demonstrated that feature-based harmonization produces stable, high-quality classification results, particularly when the chosen representation retains a rich set of informative features.

Parallel to these main research focuses, additional papers, including [**P7**] and [**P10**], investigated the influence of object distribution and diversity on classification outcomes. These works provided insight into how the proportion of overlapping objects across datasets affects model quality and demonstrated that the proposed approaches remain surprisingly robust even when local datasets share only a small fraction of common instances. Such findings help define the operational limits of dispersed classification and clarify when and why certain methods remain effective. In all, this study positively addressed all research questions, confirming that neural network-based methods across the three novel approaches achieve satisfactory classification accuracy. Both MLP and RBF networks performed well, with the RBF based method emerging as the better of the two. Imputing missing attributes across local datasets and aggregating trained weights into a global model improved classification accuracy, as did constructing a global model from

features extracted via PCA, SVD, and UMAP. These results verify our central hypothesis: neural network architectures significantly enhance the accuracy and robustness of classification in dispersed data environments.

Today, large-scale, distributed information systems are central to data processing in many domains. However, classical machine-learning techniques often remain ill-suited for real-world environments characterized by feature inconsistency, strict privacy requirements and fragmented data. The contributions presented in this thesis collectively offer a set of novel approaches that overcome these barriers. By proposing methods for prediction fusion, parameter aggregation and feature-space unification, and by validating them across a wide range of dispersed scenarios, the research expands the practical applicability of neural networks to non-uniform, privacy-sensitive settings. These advancements open new directions for decentralized learning and demonstrate that accurate global models can be constructed without violating data locality, thus significantly broadening the usability of machine-learning techniques in modern distributed systems.

Bibliography

[1] Health Insurance Portability and Accountability Act of 1996. https://www.govinfo.gov/content/pkg/PLAW-104publ191/pdf/PLAW-104publ191.pdf (1996), public Law 104-191, U.S. Statutes at Large, Vol. 110, pp. 1936–2103

[2] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016. https://eur-lex.europa.eu/eli/reg/2016/679/oj (2016), official Journal of the European Union, L 119, 4 May 2016, pp. 1–88

[3] Dry Bean. UCI Machine Learning Repository (2020), DOI: https://doi.org/10.24432/C50S4B

[4] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. p. 1175–1191. CCS '17, Association for Computing Machinery, New York, NY, USA (2017), https://doi.org/10.1145/3133956.3133982

[5] Briggs, C., Fan, Z., Andras, P.: Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–9 (2020)

[6] Caldas, S., Konečny, J., McMahan, H.B., Talwalkar, A.: Expanding the reach of federated learning by reducing client resource requirements (2019), https://arxiv.org/abs/1812.07210

[7] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. J. Artif. Int. Res. 16(1), pp. 321–357 (Jun 2002)

[8] Cipar, J., Ho, Q., Kim, J.K., Lee, S., Kumar, A., Gibbons, P., Ganger, G., Gibson, G., Xing, E.P.: Solving the straggler problem with bounded staleness. In: Proceedings of the 14th Workshop on Hot Topics in Operating Systems (HotOS). pp. 22–27 (2013)

[9] Colonna, Juan, N.E.C.M., Gordo, M.: Anuran Calls (MFCCs). UCI Machine Learning Repository (2015), DOI: https://doi.org/10.24432/C5CC9H

[10] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Le, Q.V., Mao, M.Z., Ranzato, M., Senior, A., Tucker, P., et al.: Large scale distributed deep networks. Advances in Neural Information Processing Systems (NeurIPS) 25 (2012)

[11] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)

[12] Deng, L.: The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine 29(6), pp. 141–142 (2012)

[13] Dey, R., Salem, F.M.: Gate-variants of gated recurrent unit (gru) neural networks. In: 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS). pp. 1597–1600 (2017)

[14] Dua, D., Graff, C.: Uci machine learning repository (2017), http://archive.ics.uci.edu/ml

[15] Dyczkowski, K., Pękala, B., Szkoła, J., Wilbik, A.: Federated learning with uncertainty on the example of a medical data. In: 2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). pp. 1–8 (2022)

[16] Feldman, A.M.: Majority Voting, pp. 161–177. Springer US, Boston, MA (1980), https://doi.org/10.1007/978-1-4615-8141-3_10

[17] Frank, E., Hall, M., Holmes, G., Kirkby, R., Pfahringer, B., Witten, I.H., Trigg, L.: Weka-A Machine Learning Workbench for Data Mining, pp. 1269–1277. Springer US, Boston, MA (2010), https://doi.org/10.1007/978-0-387-09823-4_66

[18] Gerbessiotis, A., Valiant, L.: Direct bulk-synchronous parallel algorithms. Journal of Parallel and Distributed Computing 22(2), pp. 251–267 (1994), https://www.sciencedirect.com/science/article/pii/S0743731584710859

[19] Geyer, R.C., Klein, T., Nabi, M.: Differentially private federated learning: A client level perspective (2018), https://arxiv.org/abs/1712.07557

[20] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. 9(8), pp. 1735–1780 (Nov 1997), https://doi.org/10.1162/neco.1997.9.8.1735

[21] Johnson, B.: Crowdsourced Mapping. UCI Machine Learning Repository (2016), DOI: https://doi.org/10.24432/C56315

[22] Kall, S., Trabelsi, S.: An asynchronous federated learning approach for a security source code scanner. In: Proceedings of the 7th International Conference on Information Systems Security and Privacy - ICISSP. pp. 572–579. INSTICC, SciTePress (2021)

[23] Klema, V., Laub, A.: The singular value decomposition: Its computation and some applications. IEEE Transactions on Automatic Control 25(2), pp. 164–176 (1980)

[24] Koonce, B.: ResNet 50, pp. 63–72. Apress, Berkeley, CA (2021), https://doi.org/10.1007/978-1-4842-6168-2_6

[25] Krizhevsky, A.: Learning multiple layers of features from tiny images. University of Toronto (05 2012)

[26] Kuncheva, L.I., Bezdek, J.C., Duin, R.P.: Decision templates for multiple classifier fusion: an experimental comparison. Pattern Recognition 34(2), pp. 299–314 (2001), https://www.sciencedirect.com/science/article/pii/S003132039900223X

[27] Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. In: Proceedings of Machine Learning and Systems (MLSys) (2020), https://proceedings.mlsys.org/paper/2020/file/1f5fe83998a09396ebe6477d9475ba0c-Paper.pdf

[28] Li, X., Jiang, M., Zhang, X., Kamp, M., Dou, Q.: Fedbn: Federated learning on non-iid features via local batch normalization (2021), https://arxiv.org/abs/2102.07623

[29] Lin, Y., Han, S., Mao, H., Wang, Y., Dally, W.J.: Deep gradient compression: Reducing the communication bandwidth for distributed training. In: International Conference on Learning Representations (ICLR) (2018)

[30] Liu, D.C., Nocedal, J.: On the limited memory bfgs method for large scale optimization. Mathematical Programming 45(1–3), pp. 503–528 (Aug 1989), https://doi.org/10.1007/BF01589116

[31] Liu, L., Zhang, J., Song, S., Letaief, K.B.: Client-edge-cloud hierarchical federated learning. In: ICC 2020-2020 IEEE international conference on communications (ICC). pp. 1–6. IEEE (2020)

[32] Lloyd, S.: Least squares quantization in pcm. IEEE Transactions on Information Theory 28(2), pp. 129–137 (1982)

[33] Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of English: The Penn Treebank. Computational Linguistics 19(2), pp. 313–330 (1993), https://aclanthology.org/J93-2004/

[34] Maćkiewicz, A., Ratajczak, W.: Principal components analysis (pca). Computers and Geosciences 19(3), pp. 303–342 (1993), https://www.sciencedirect.com/science/article/pii/009830049390090R

[35] McInnes, L., Healy, J., Saul, N., Großberger, L.: Umap: Uniform manifold approximation and projection. Journal of Open Source Software 3(29), pp. 861 (2018), https://doi.org/10.21105/joss.00861

[36] McMahan, H.B., Moore, E., Ramage, D., Hampson, S., et al.: Communication-efficient learning of deep networks from decentralized data. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS) pp. 1273–1282 (2017), https://proceedings.mlr.press/v54/mcmahan17a/mcmahan17a.pdf

[37] Melis, L., Song, C., De Cristofaro, E., Shmatikov, V.: Exploiting unintended feature leakage in collaborative learning. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 691–706 (2019)

[38] Michalski, R., Chilausky, R.: Soybean (Large). UCI Machine Learning Repository (1980), DOI: https://doi.org/10.24432/C5JG6Z

[39] Miikkulainen, R.: Simple Recurrent Network, pp. 906–906. Springer US, Boston, MA (2010), https://doi.org/10.1007/978-0-387-30164-8_762

[40] Mowforth, P., Shepherd, B.: Statlog (Vehicle Silhouettes). UCI Machine Learning Repository, DOI: https://doi.org/10.24432/C5HG6N

[41] Nishio, T., Yonetani, R.: Client selection for federated learning with heterogeneous resources in mobile edge. In: ICC 2019 - 2019 IEEE International Conference on Communications (ICC). pp. 1–7 (2019)

[42] Panayotov, V., Chen, G., Povey, D., Khudanpur, S.: Librispeech: An asr corpus based on public domain audio books. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 5206–5210 (2015)

[43] Popescu, M.C., Balas, V.E., Perescu-Popescu, L., Mastorakis, N.: Multilayer perceptron and neural networks. WSEAS Trans. Cir. and Sys. 8(7), pp. 579–588 (Jul 2009)

[44] Reddi, S., Charles, Z.B., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., McMahan, B. (eds.): Adaptive Federated Optimization (2021), https://openreview.net/forum?id=LkFG3lB13U5

[45] Reisizadeh, A., Mokhtari, A., Hassani, H., Jadbabaie, A., Pedarsani, R.: Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In: International Conference on Artificial Intelligence and Statistics (AISTATS). pp. 2021–2031 (2020)

[46] Roy, C., Chakraborty, D., Debnath, S., Mukherjee, A., Chaki, N.: Single failure recovery in distributed social network. In: Hong, T., Wojtkiewicz, K., Chawuthai, R., Sitek, P. (eds.) Recent Challenges in Intelligent Information and Database Systems (ACIIDS 2021), Communications in Computer and Information Science, vol. 1371, pp. 203–215. Springer (2021), https://link.springer.com/chapter/10.1007/978-981-16-1685-3_17

[47] Saari, D.G.: Selecting a voting method: the case for the borda count. Constitutional Political Economy 34(3), pp. 357–366 (2023), https://doi.org/10.1007/s10602-022-09380-y

[48] Sanders, J., Kandrot, E.: CUDA by Example: An Introduction to General-Purpose GPU Programming. Addison-Wesley Professional, 1st edn. (2010)

[49] SAP: credential-digger: A github scanning tool that identifies hardcoded credentials. GitHub repository, version v4.13.0 (Dec 2023), https://github.com/SAP/credential-digger

[50] Seide, F., Fu, H., Droppo, J., Li, G.: 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. pp. 1058–1062. Interspeech (2014), https://doi.org/10.21437/Interspeech.2014-274

[51] Srinivasan, A.: Statlog (Landsat Satellite). UCI Machine Learning Repository (1993), DOI: https://doi.org/10.24432/C55887

[52] Strumiłło, P., Kamiński, W.: Radial basis function neural networks: Theory and applications. In: Rutkowski, L., Kacprzyk, J. (eds.) Neural Networks and Soft Computing. pp. 107–119. Physica-Verlag HD, Heidelberg (2003)

[53] Tan, Q., Wu, S., Tao, Y.: Privacy-enhanced federated learning for non-iid data. Mathematics 11(19) (2023), https://www.mdpi.com/2227-7390/11/19/4123

[54] Zhu, L., Liu, Z., Han, S.: Deep leakage from gradients. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019), https://proceedings.neurips.cc/paper_files/paper/2019/file/60a6c4002cc7b29142def8871531281a-Paper.pdf

[55] Zwitter, M., Soklic, M.: Lymphography. UCI Machine Learning Repository (1988), DOI: https://doi.org/10.24432/C54598

Publication reprints

Publication [P1]

Przybyła-Kasperek M., Marfo K.F. Neural network used for the fusion of predictions obtained by the k-nearest neighbors algorithm based on independent data sources. *Entropy*, 23(12), 2021. doi:10.3390/e23121568
URL: http://dx.doi.org/10.3390/e23121568.
DOI: 10.3390/e23121568.
$\text{MEiN}_{2021} = 100$
Number of citations:

- according to Web of Science: 11
- according to Google Scholar: 16

Contributor	Description of main tasks
Małgorzata Przybyła-Kasperek	- conceptualization and methodology - result analysis - manuscript: original draft preparation - manuscript: review and editing - visualization: creating all figures in manuscript
Kwabena Frimpong Marfo	- investigation - methodology - result analysis - implementation of proposed algorithm - manuscript: review and editing - visualization: creating all figures in manuscript

entropy

MDPI

Article

Neural Network Used for the Fusion of Predictions Obtained by the K-Nearest Neighbors Algorithm Based on Independent Data Sources

Małgorzata Przybyła-Kasperek * and Kwabena Frimpong Marfo

Institute of Computer Science, Faculty of Science and Technology, University of Silesia in Katowice, Będzińska 39, 41-200 Sosnowiec, Poland; kmarfo@us.edu.pl

* Correspondence: malgorzata.przybyla-kasperek@us.edu.pl; Tel.: +48-32-269-17-56

Abstract: The article concerns the problem of classification based on independent data sets—local decision tables. The aim of the paper is to propose a classification model for dispersed data using a modified k-nearest neighbors algorithm and a neural network. A neural network, more specifically a multilayer perceptron, is used to combine the prediction results obtained based on local tables. Prediction results are stored in the measurement level and generated using a modified *k*-nearest neighbors algorithm. The task of neural networks is to combine these results and provide a common prediction. In the article various structures of neural networks (different number of neurons in the hidden layer) are studied and the results are compared with the results generated by other fusion methods, such as the majority voting, the Borda count method, the sum rule, the method that is based on decision templates and the method that is based on theory of evidence. Based on the obtained results, it was found that the neural network always generates unambiguous decisions, which is a great advantage as most of the other fusion methods generate ties. Moreover, if only unambiguous results were considered, the use of a neural network gives much better results than other fusion methods. If we allow ambiguity, some fusion methods are slightly better, but it is the result of this fact that it is possible to generate few decisions for the test object.

Keywords: neural network; fusion method; independent data sources; *k*-nearest neighbors algorithm; dispersed data

Citation: Przybyła-Kasperek, M.; Marfo, K.F. Neural Network Used for the Fusion of Predictions Obtained by the K-Nearest Neighbors Algorithm Based on Independent Data Sources. *Entropy* **2021**, *23*, 1568. https://doi.org/10.3390/e23121568

Academic Editor: Sotiris Kotsiantis

Received: 27 October 2021
Accepted: 23 November 2021
Published: 25 November 2021

1. Introduction

The article is devoted to the issue of classification based on dispersed data. More precisely, data collected in many local decision tables, which were provided by independent units, are considered. This approach is considered for example in federated learning [1,2] and fog computing [3] approaches.

The considered approach differs from an ensemble classifiers approach mainly due to the form of data that is considered. In ensemble learning, a set of local tables is generated based on one decision table in a controlled manner. Thus, we can define the form of attributes' sets and objects' sets in local tables, and that simplifies a lot. It is completely different in the case of dispersed data. The set of local decision tables is already collected independently, and we have no influence on its form. In addition, sets of attributes in local tables can be quite different, but some elements may be common between sets. A similar property applies to sets of objects. In addition, we cannot expect that we have universal identifiers for objects in all local tables, so it is impossible to verify which objects are common between local tables.

In real applications, very often, data are collected in such a dispersed way. In specialized departments located in various hospitals, various banks, or even on smart phones—information about the user. Each application installed locally can collect a personalized local table. The domain of such dispersed data can be very different. However, the general

problem remains the same—how to use dispersed data efficiently considering all inconsistent local tables at the same time. This is very significant and one of the top problems in today's multi-devices world.

This leads to completely different considerations, causes inconsistencies and conflicts between local tables. In addition, there are very important issues regarding data protection and data privacy [4]. We do not deal with this problem in the publication; however, we follow all the rules by not sharing raw data, only prediction results in central domain.

A common approach to deal with dispersed data is to build a separate local model based on each local table and then combine the local prediction results [5–7]. In the stage of combining the prediction results, we can use fusion methods [8] from three different levels (measurement level, rank level and abstract level). Another approach is to build a meta-model that will train on how to generate global results based on predictions obtained from local models.

Fusion methods very often generate ambiguous results in the case of dispersed data (ties occur). The motivation of the study is to propose a model that will generate unambiguous results for dispersed data. We know that neural networks can learn to recognize even multilayer and complex patterns very well. Prediction results generated by local tables (especially from the measurement level) are often contradictory, ambiguous and it is difficult to generate a global decision based on them. Therefore, neural networks seem to be the appropriate approach to working with such data.

This paper proposes the use of a neural network in conjunction with a modified *k*-nearest neighbors algorithm for classification problems based on dispersed data. In the literature, we can find applications of neural networks in the stage of predictions fusion [9–11]. These applications are in specialized fields and rely on the use of completely different methods to generate local predictions than those proposed in this paper.

The contribution of this paper is to propose a classification model for dispersed data using a modified *k*-nearest neighbors algorithm and a neural network. The first step in the model is to use the *k*-nearest neighbors algorithm to generate prediction vectors based on local tables. Prediction vectors are defined over the decision classes. Therefore, it was required to modify this algorithm due to the need to generate certainty that the object belongs to each specific decision class. This algorithm ensures that the prediction vector will be determined based on the most relevant objects to the currently considered test object. In the next stage, it is proposed to use a neural network to generate the final decision. The network is trained to make the correct decision based on such prediction vectors. It is not an easy task, because total agreement is very unlikely to be obtained in the prediction vectors. Particular systems of values in vectors may indicate a specific decision value. It may happen that some local tables are better in recognizing objects from a specified subspace. Neural networks seem to be an appropriate model for recognizing all these complex relationships.

The paper is organized as follows. In Section 2, the proposed classification model using dispersed data is described. Each of the stages in the model—generating local predictions and combining local predictions is presented in a separate subsection. The algorithm's description and the discussion about the computational complexity are given. Then, a graphical presentation of the method and an overall overview are presented. In the last subsection, other fusion methods that are used in this article as the baseline method are also discussed. Section 3 addresses the data sets that were used and presents the conducted experiments and discussion on obtained results. Section 4 is on conclusions and future research plans.

2. Materials and Methods

This section describes the classification model for dispersed data available in many local decision tables. This classification process consists of two stages:

- At the first stage, a model is built based on each local decision table. Using a modified *k*-nearest neighbors algorithm, predictions from the measurement level are designated.

- At the second stage, a neural network is used to aggregate the predicted results to determine the final classification.

2.1. The First Stage in a Dispersed Classification Model

We assume that a set of decision tables is given. The tables were collected independently by separate units. Based on each table, a classifier is built. We assume that a set of decision tables $D_{ag} = (U_{ag}, A_{ag}, d)$, $ag \in Ag$ from one discipline is available, where U_{ag} is the universe, a set of objects; A_{ag} is a set of conditional attributes; d is a decision attribute. Decision tables are collected independently, so both sets of objects and sets of attributes can have any form. They can have common elements between tables, but do not have to. The only condition that must be met by local tables is collecting data from one discipline. Formally, this is satisfied by the assumption that the same decision attribute is present in all tables. Aggregation for such tables is difficult and can generate inconsistencies, therefore aggregation is not performed, but predictions are made separately based on each table.

Ag is a set of classifiers and an identifier $ag \in Ag$ is a classifier that is built based on a decision table D_{ag}. In general, any classifier may be used for this purpose. In this study, the modified k-nearest neighbors classifier with the Gower similarity measure was used. For each local table and for each test object x, a probability vector over decision classes (denoted by $\mu_{ag}(x)$) is designated. The dimension of vectors $\mu_{ag}(x) = [\mu_{ag,1}(x), \ldots, \mu_{ag,c}(x)]$ is equal to the number of decision classes $c = card\{V^d\}$, where V^d is a set of decision attribute values from all decision tables and $card\{V^d\}$ is the cardinality of this set. Each coefficient $\mu_{ag,j}(x)$ is determined using the k-nearest neighbors of the test object x belonging to a given decision class j and decision table D_{ag}. The Gower similarity measure [12] is used because it allows the analysis of data with different types of attributes and data with missing values. At the end of this stage, we obtain for each test object a set of vectors over the decision classes in a cardinality equal to the number of local tables.

In this way, we obtain local predictions, but not the final global decision. The k-nearest neighbors method is used for the computation of local predictions, because this method is not computationally complex and easily scalable even for large and multidimensional data. Moreover, in the k-nearest neighbors algorithm, predictions rely strictly on the relevant objects from the data sets. Thanks to this, we obtain diversified classifiers for dispersed data (as local tables are independent), which, as we know from the literature [13], is important for an ensemble of classifiers. In addition, the k-nearest neighbors method has already been used in previous studies for dispersed data [14,15] and has produced good results. To generate the final global decision, we propose training a neural network in making the final decision using the probability vectors generated based on local tables.

The pseudo-code of the algorithm that generates predictions based on local tables is given in Algorithm 1.

The computational complexity of the above algorithm is rather small and equal to $O(card\{Ag\} \times \max_{ag \in Ag} card\{U_{ag}\} \times card\{U_{test}\})$.

2.2. The Second Stage in a Dispersed Classification Model

As the results of the previous stage, we obtain a set of vectors over the decision classes $\mu_{ag}(x), x \in U_{test}, ag \in Ag$. However, we do not have a final decision for test objects. To determine a global decision for the test object x, we must fuse the vectors $\mu_{ag}(x), ag \in Ag$. We propose the use of a neural network for this purpose. More formally, a multilayer perceptron is applied. We use one network structure. This network consists of three layers: an input, a hidden layer and an output layer. The input of the neural network are the values of vectors generated in the previous step of classification. The number of neurons in the input layer is equal to the number of local decision tables multiplied by the number of decision classes $card\{Ag\} \times card\{V^d\}$. Thus, the greater the data dispersion, the greater the complexity. The number of neurons in the output layer is equal to the number of decision classes. Each of the neurons determines the probability with which the test object belong to a given decision class.

Algorithm 1 Pseudo-code of algorithm generating predictions based on local tables

Input: A set of local decision tables $D_{ag} = (U_{ag}, A_{ag}, d), ag \in Ag$; test set—a decision table $D_{test} = (U_{test}, A_{test}, d_{test})$, $A_{test} = \bigcup_{ag \in Ag} A_{ag}$

Output: A set of vectors over decision classes $\mu_{ag}(x), ag \in Ag, x \in U_{test}$.

foreach $x \in U_{test}$

  foreach $ag \in Ag$

    Calculate the value of the Gower similarity measure for each object in the set U_{ag} and the test object x (using only the attributes from the set A_{ag}).

    For each j-th decision class, find the k nearest neighbors objects to x from j-th decision class of the decision table D_{ag}.

    $\mu_{ag,j}(x)$ is equal to the mean of similarity of the k nearest neighbors objects from the j-th decision class of the decision table D_{ag}.

  end foreach

end foreach

In this study, different numbers of neurons in the hidden layer are analyzed. Such a number should depend on the number of neurons in the input layer, and this is dependent on the data. Therefore, the following values will be analyzed: $\{1, 3, 4, 4.25, 4.5, 4.75, 5\} \times$ the number of neurons in the input layer. For the hidden layer, the ReLU (Rectified Linear Unit) activation function is used, as it is the most popular activation function and gives very good results [16]. For the output layer, the SoftMax activation function is used, which is recommended when we deal with a multi-class problem [17]. In this paper, data sets containing from four to 19 decision classes are analyzed. The neural network is trained using the backpropagation method. A gradient descent method, with an adaptive step size is used in the backpropagation method. It is known that the SoftMax layer give good results with the Adam optimizer [18]. The Adam optimizer was proposed in [19] and is one of the most popular adaptive step size methods. From [20] we know that the categorical cross-entropy loss gives best results with SoftMax layer. That is why the Adam optimizer and the categorical cross-entropy loss function are used in the study.

The code that is used defines a neural network using Keras library in Python.

The pseudo-code of the algorithm that defines a neural network with one hidden layer is given in Listing 1.

Listing 1. Neural network with one hidden layer.

```
def neural_model(inputDim:int, neurons: int):
    model = Sequential()
    model.add(Dense(neurons, input_dim=inputDim,
                    activation='relu'))
    model.add(Dense(classes, activation='softmax'))
    model.compile(loss='categorical_crossentropy',
                  optimizer='adam', metrics=['accuracy'])
return model
```

The neural network is trained based on the test set. It should be emphasized that the objects from the training set are not used for this purpose. The objects from the training set are used to determine the probability vectors, i.e., the k-nearest neighbors classifier is used for the training set. Thus, to study the classification quality of neural network, the 10-fold cross-validation method is used for test set. Thus, each time nine folds from the test set are used to train the neural network, while the last fold is used to analyze the classification quality.

Figure 1 presents the described above stages in a dispersed classification model.

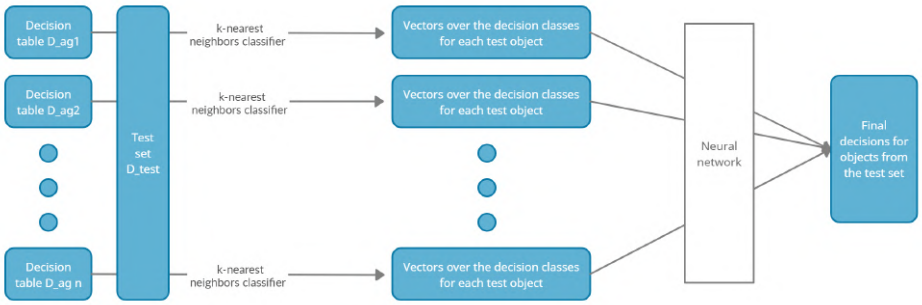


Figure 1. Stages in the dispersed classification model.

In the proposed model, two main algorithms are used. A modified *k*-nearest neighbors algorithm and a neural network. The first one has low computational complexity and classifies test objects based on dispersed local tables. Due to the dispersion of data and the differences in the attributes sets in local tables, it is not possible to generate an aggregated result using this algorithm. Therefore, it is necessary to use a different method of designating the global decision. For this purpose, a neural network is used. The training process of the neural network is a complex process. However, it should be remembered that it is only implemented once. The process of test objects' classification in the neural network is a task with very low computational complexity. It consists only of performing arithmetic operations in the number depending on the number of layers in the network and the number of neurons in each layer. The great advantage of neural network used as a fusion method is obtaining unambiguous results. This approach does not generate ties—one decision class is generated always.

2.3. Other Fusion Methods

The results obtained using the proposed method are compared with the results generated by other fusion methods known from the literature. These methods can be used to aggregate the predictions in the form of probability vectors. Thus, a modified *k*-nearest neighbors algorithm is used as described above. However, instead of a neural network, the results are aggregated using the fusion method.

For this purpose, five methods were selected. The chosen methods are from different measurement groups, they are characterized by completely different criteria and have different degrees of complexity. Three of the methods are simple and very popular. The other two methods are much more computationally complex and sophisticated.

The first method from the measurement level is the sum rule. The sum rule consists of the designation of the sum of the probability values assigned to one decision class by each of local tables. The set of decisions that have the maximum of these sums is the final decision

$$\arg \max_{j \in \{1,\ldots,c\}} \Big\{ \sum_{ag \in Ag} \mu_{ag,j}(x) \Big\}.$$

The next fusion method, the Borda count method, belongs to the rank level, which means that the ranks vectors are generated based on the probability vectors. Ranks are assigned within vectors in such a way that decision classes that have greater probability also have higher rank. The method consists of designating the sum of the number of classes ranked below the given class by each local decision table. Thus, the value is determined for each decision class j $\sum_{ag \in Ag} \Big(card\{V^d\} - r_{ag,j}(x) \Big)$, where $r_{ag,j}(x)$ is the rank assigned based on the probability value $\mu_{ag,j}(x)$. The set of classes that have the maximum value of the Borda count is the final decision.

The third fusion method that is used is the majority voting method and belongs to the abstract level. In this method, each local table gives one vote for each decision that has the maximum probability in the probability vectors. Final decisions are those which received the maximum number of votes.

The next two methods are much more complex and sophisticated, both require a training stage.

The method that is based on decision templates belongs to the measurement level and was proposed in the paper [21]. The method uses decision profile, i.e., the matrix with dimension number of local tables $\times$ number of decision classes. The rows of the matrix are the probability vectors from the measurement level generated based on local tables. This is the way of presenting the local tables outputs for any object. For each class, the decision templates are defined based on the decision profiles that are constructed for the objects from the training set. The decision template DT_j for class j is the average of the decision profiles of the objects of the training set in class j. These decision templates can be seen as patterns for decision classes and the process of determining them is the training process of the method. When a test object is considered, first a decision profile for the object is determined, then, the similarity between the decision profile for the test object and the decision template for each class is calculated. For this purpose, one of the distance measures is used. The Euclidean distance, the Hamming distance, the Jaccard similarity or the Symmetric difference are usually applied. The set of classes that have the maximum value of similarity is the final decision. In this article, the Hamming distance is used, because based on previous experience [22] it produces the best results.

The method that is based on the theory of evidence belongs to the measurement level and was proposed in the paper [23]. In this method as in the previous method, the decision templates for decision classes $DT_j, j \in \{1, \ldots, c\}$ are designated based on the training set. The decision templates and the decision profile for the test object are compared using the Dempster-Shafer theory and the belief is calculated. The following steps are performed in the Dempster-Shafer algorithm:

1. Let $DT_j(m, \cdot)$ denote the m-th row of the decision template for class j and $DP_{m,\cdot}(x)$ denote the m-th row of the decision profile for the object x. The proximity between the prediction calculated based on the m-th local table $DP_{m,\cdot}(x)$ and the m-th row of the decision template for every class $j \in \{1, \ldots, c\}$ and for each local table m is calculated

$$\phi_{j,m}(x) = \frac{(1 + \left\| DT_j(m, \cdot) - DP_{m,\cdot}(x) \right\|^2)^{-1}}{\sum_{k=1}^{c} (1 + \left\| DT_k(m, \cdot) - DP_{m,\cdot}(x) \right\|^2)^{-1}}$$

   where $\|\cdot\|$ is the norm. The Euclidean norm was applied in this study.
2. For every class $j \in \{1, \ldots, c\}$ and for each local table m the following belief degrees are calculated

$$Bel_j(DP_{m,\cdot}(x)) = \frac{\phi_{j,m}(x) \prod_{k \neq j} (1 - \phi_{k,m}(x))}{1 - \phi_{j,m}(x)[1 - \prod_{k \neq j} (1 - \phi_{k,m}(x))]}.$$

3. The Dempster-Shafer membership degrees for every class $j \in \{1, \ldots, c\}$ are calculated

$$\mu_j(x) = K \prod_m Bel_j(DP_{m,\cdot}(x))$$

   where K is a constant that ensures that $\mu_j(x) \leq 1$.

The set of classes that have the maximum value of the Dempster-Shafer membership degrees is the final decision.

All the above-mentioned fusion methods can generate ties. In such a situation, instead of one final decision, a set of decisions is generated. In this article, these ties are not solved. In the next section, two measures of classification quality are analyzed to compare the ambiguity of results.

3. Results

The experiments were conducted with the data taken from the UC Irvine Machine Learning Repository and one artificially generated data. Each data available in the repos-

itory is stored in a single table. Before the experiments were performed, the data were dispersed. For this reason, not all data sets could be used for analysis. An important issue is the presence of many conditional attributes in the data. Moreover, the proposed model uses the Gower measure, which is dedicated to data with different types of attributes (qualitative, quantitative, binary). Therefore, it is important that the attributes in the data set are of different types. The existence of multi-decision classes in a data set is also an important matter, as some of the fusion methods used for comparison may generate ties. Ties in the case of a small number of decision classes (in extreme cases two decision classes) are not acceptable.

Three data sets meeting the above conditions were selected for the analysis: the Lypmhography, the Vehicle Silhouettes and the Soybean (Large) data sets. For the Soybean data set, the two independent data sets: a test set and a training set are available in the repository. The Vehicle and the Lymphography data sets were randomly divided into two disjoint subsets, the training set (70% of objects) and the test set (30% of objects). This was done to apply the same testing strategy for all analyzed data sets (train and test method).

The artificial data were generated using the Weka software [24]. For this purpose, the function RandomRBF was used. This function, at first, randomly generates centers for each decision class. Then to each center, a weight is randomly assigned and a central point per attribute, and a standard deviation. The new object is generated as follows. The center is selected—according to the weights. Attribute values are randomly generated and offset from the center. Then the vector is scaled so that its length equals a value sampled randomly from the Gaussian distribution of the center. The decision class is assigned based on the center. The following settings were used:

- Number of numerical conditional attributes—30
- Number of objects—399
- Number of decision classes—7
- Number of centroids (To make the set more difficult, several centroids were used to generate objects from one decision class)—50
- Seed for generating random numbers—1

Thus, unbalanced data were obtained, the numbers of objects in individual decision classes are as follows: $21, 36, 31, 57, 58, 107, 89$. The data set was randomly divided into two disjoint subsets, the training set (299 objects) and the test set (100 objects) in a stratified mode. Additionally, noise has been added to each attribute in the training set. For this purpose, the values generated from the normal distribution with an average of 0 and a standard deviation of 0.3 were used and added to each value in the training set.

For dispersed data, the cross-validation method is too complex. Moreover, the cross-validation method would be difficult to apply because the test set should contain objects that have defined values on all the conditional attributes that belong to local tables. In addition, this restricts, and even makes it impossible to draw independently from local tables. Data characteristics are given in Table 1.

Table 1. Data set characteristics.

Data Set	# The Training Set	# The Test Set	# Conditional Attributes	# Decision Classes
Vehicle Silhouettes	592	254	18	4
Lymphography	104	44	18	4
Soybean	307	376	35	19
Artificial Data	299	100	30	7

The training sets of the above-mentioned data sets were dispersed. To check for different degrees of dispersion for each data set, five different dispersed versions with 3, 5, 7, 9 and 11 local tables were prepared. Conditional attributes for local tables were selected randomly, but each local table contained only a small subset of the full set of attributes.

Certain attributes were common to several local tables. The decision attribute was included in each of the tables. The full set of objects is also stored in each of the local tables, but without identifiers. This reflects the real situation where we cannot identify the objects between the tables.

The quality of classification was evaluated based on the test set. Three measures were analyzed. The first measure is the estimator of classification error e. It is a fraction of the total number of objects in the test set that were classified incorrectly. An object is considered to be correctly classified when its correct decision class belongs to the generated decision set. The second measure is the estimator of classification ambiguity error e_{ONE}. It is also a fraction of the total number of objects in the test set that were classified incorrectly. However, this time an object is considered to be correctly classified when only one correct decision class was generated. More strictly, this measure does not accept ambiguity. The third measure is the average number of generated decisions sets $\bar{d}$. The third measure allows an assessment of how often and how numerous are the draws generated by the dispersed classification model and the fusion methods.

It should be noted once again that to use the neural network, a 10-fold cross-validation was used on the test set, i.e., the neural network was trained 10 times with 9 folds and tested on one remaining fold. In addition, each test was performed three times to ensure that the results were reliable and not distorted by the influence of randomness. The results for the neural network approach that are given below are the average of the obtained results.

The experiments were carried out according to the following scheme:

- Generating vectors of predictions based on local tables using the k-nearest neighbors classifier. For each data set, three different values of the k parameter were tested, namely $k \in \{1, 5, 10\}$. One parameter value was selected for each data set that produced the best overall results. For the Vehicle Silhouettes data set—$k = 5$, for the Lymphography data set—$k = 10$, for the Soybean data set—$k = 1$, for the artificial data set—$k = 1$ were selected.
- Generating a global decision using a neural network with one hidden layer and different number of neurons in the hidden layer. For each data set, the following number of neurons in the hidden layer were tested: $\{1, 3, 4, 4.25, 4.5, 4.75, 5\}\times$ the number of neurons in the input layer. Different number of neurons in the hidden layer was also checked. However, it was noticed that the accuracy of the respective models improves as the number of neurons in the hidden layer increases, but significant improvement declines around $5\times$ the number of neurons in the input layer. The number of neurons in the input layer depends on the number of local tables. Thus, the more dispersed data we have, the more complex the structure of the neural network is.
- Generating a global decision using one of the fusion methods: the sum rule, the Borda count, the majority vote, the method that is based on decision templates and the method that is based on theory of evidence.

Comparison of experimental results was made in terms of:

- The quality of classification for different numbers of neurons in the hidden layer;
- The quality of classification of the proposed dispersed classification model vs. other fusion methods (the sum rule, the Borda count, the majority vote, the method that is based on decision templates and the method that is based on theory of evidence).

3.1. Comparison of Experimental Results for Different Numbers of Neurons in the Hidden Layer

Table 2 presents average classification error e, average classification ambiguity error e_{ONE} and the average number of generated decisions set $\bar{d}$ for all dispersed data sets and the neural network approach with different number of neurons in the hidden layer. Using 10-fold cross-validation method, experiments were repeated 3 times. The minimal mean errors have been marked in bold. A very important thing that should be noticed is that all generated results are unambiguous. Always a single decision class is generated. So, for the proposed model we have the property $e = e_{ONE}$ and $\bar{d} = 1$.

Table 2. Results of classification error e, classification ambiguity error e_{ONE} and the average number of generated decisions $\bar{d}$ for the dispersed system with neural network. Designation #Input is used for the number of neurons in the input layer.

Data Set	No. on Local Tables	No. of Neurons in Hidden Layer						
		1 × #Input $e = e_{ONE}/\bar{d}$	**3 × #Input** $e = e_{ONE}/\bar{d}$	**4 × #Input** $e = e_{ONE}/\bar{d}$	**4.25 × #Input** $e = e_{ONE}/\bar{d}$	**4.5 × #Input** $e = e_{ONE}/\bar{d}$	**4.75 × #Input** $e = e_{ONE}/\bar{d}$	**5 × #Input** $e = e_{ONE}/\bar{d}$
Lypmho graphy	3	0.286/1	0.281/1	**0.246**/1	0.259/1	0.251/1	0.288/1	0.273/1
	5	**0.194**/1	0.251/1	0.258/1	0.264/1	0.253/1	0.258/1	0.259/1
	7	0.326/1	0.328/1	**0.308**/1	0.316/1	0.314/1	0.321/1	0.339/1
	9	0.278/1	0.256/1	0.256/1	**0.249**/1	**0.249**/1	0.271/1	**0.249**/1
	11	**0.244**/1	0.278/1	0.276/1	0.269/1	0.276/1	0.271/1	0.294/1
Vehicle	3	0.365/1	0.296/1	0.279/1	0.283/1	0.284/1	0.096/1	0.089/1
	5	0.331/1	0.284/1	**0.278**/1	**0.278**/1	0.294/1	0.296/1	0.291/1
	7	0.326/1	0.299/1	0.291/1	**0.281**/1	0.304/1	0.284/1	0.298/1
	9	0.298/1	0.284/1	**0.275**/1	0.293/1	0.307/1	0.314/1	0.295/1
	11	0.327/1	0.303/1	**0.274**/1	0.293/1	0.284/1	0.315/1	0.302/1
Soybean	3	0.093/1	0.092/1	0.091/1	**0.084**/1	0.085/1	0.096/1	0.089/1
	5	0.088/1	0.094/1	0.091/1	0.099/1	0.093/1	0.093/1	**0.085**/1
	7	0.099/1	0.093/1	0.090/1	0.089/1	**0.087**/1	0.090/1	0.093/1
	9	0.078/1	0.081/1	**0.073**/1	0.081/1	0.075/1	0.079/1	0.084/1
	11	0.068/1	0.069/1	0.068/1	0.071/1	0.074/1	**0.061**/1	0.077/1
Artificial Data	3	0.393/1	0.190/1	0.113/1	0.103/1	0.106/1	**0.090**/1	0.106/1
	5	0.193/1	0.056/1	0.060/1	**0.023**/1	0.026/1	0.033/1	0.233/1
	7	0.166/1	0.030/1	0.033/1	0.016/1	**0.010**/1	0.233/1	0.233/1
	9	0.143/1	0.036/1	0.036/1	0.033/1	0.036/1	**0.026**/1	0.033/1
	11	0.130/1	0.050/1	0.043/1	**0.036**/1	0.050/1	0.043/1	0.040/1
average e		0.221	0.183	0.172	0.171	0.173	0.187	0.197

Based on results from Table 2, the calculated means and the individual results for the dispersed data sets, it can be seen that the best results are obtained for the number of neurons in the hidden layer of approximately equal 4× the number of neurons in the input layer.

The Friedman's test was performed. All results (the number of neurons in the hidden layer: $\{1, 3, 4, 4.25, 4.5, 4.75, 5\}\times$ the number of neurons in the input layer) were selected—each number of neurons as a separate group, the test confirmed that differences among the classification error in these seven groups are significant, with a level of $p = 0.002599$. Then, to determine the pairs of groups between which statistically significant differences occur, the Wilcoxon each pair test for dependent groups were performed. The test showed that there is significant difference with $p < 0.05$ between

- Group 1 (1 × #Input) and four other groups (Group 2—3 × #Input, Group 3—4 × #Input, Group 4—4.25 × #Input and Group 5—4.5 × #Input),
- Group 2 (3 × #Input) and three other groups (Group 3—4 × #Input, Group 4—4.25 × #Input and Group 5—4.5 × #Input),
- Group 3 (4 × #Input) and Group 7—5 × #Input,
- Group 4 (4.25 × #Input) and Group 7—5 × #Input.

Additionally, comparative box-plot chart for the values of the classification error was created (Figure 2). As can be observed, distributions of the classification error values in groups are quite different—especially better results are visible for the numbers of neurons in the hidden layer equal to 4 × #Input and 4.25 × #Input.

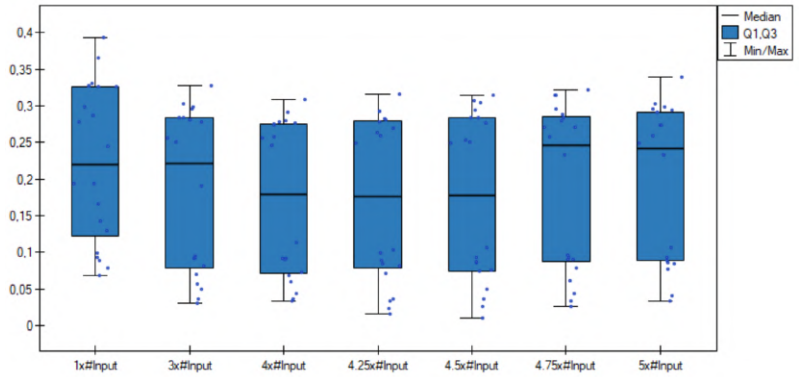


Figure 2. Box-plot chart with (Median, the first quartile—Q1, the third quartile—Q3) the value of classification error e for the neural network with different numbers of neurons in the hidden layer.

3.2. Comparison of Experimental Results for the Proposed Dispersed Classification Model versus Other Fusion Methods

Table 3 presents classification error e, classification ambiguity error e_{ONE} and the average number of generated decisions set $\bar{d}$ for all dispersed data sets and fusion methods: the sum rule, the Borda count, the majority voting, the method that is based on decision templates and the method that is based on theory of evidence. These results were obtained using train and test method. The tests were performed on the test sets on which the neural network was trained and tested at an earlier stage of the experiments. As before, the following k values were used in the modified k nearest neighbors algorithm: the Vehicle Silhouettes data set—$k = 5$, for the Lymphography data set—$k = 10$, for the Soybean data set—$k = 1$ and for the artificial data set—$k = 1$. The tests were performed only once as these fusion methods are deterministic. As can be seen, virtually all results are ambiguous. Only for the method that is based on decision templates and the method that is based on theory of evidence always one decision class is generated. Additionally for the Vehicle data set and the sum rule one decision class is generated.

Table 3. Results of classification error e, classification ambiguity error e_{ONE} and the average number of generated decisions $\bar{d}$ for fusion methods—the sum rule, the Borda count, the majority voting, the method that is based on decision templates and the method that is based on theory of evidence.

Data Set	No. of Local Tables	Sum Rule $e/e_{ONE}/\bar{d}$	Borda Count $e/e_{ONE}/\bar{d}$	Majority Vote $e/e_{ONE}/\bar{d}$	Decision Templates $e/e_{ONE}/\bar{d}$	Theory of Evidence $e/e_{ONE}/\bar{d}$
Lypmho graphy	3	0.250/0.250/1	0.159/0.386/1.227	0.136/0.409/1.273	0.159/0.159/1	0.273/0.273/1
	5	0.273/0.318/1.045	0.205/0.318/1.114	0.205/0.318/1.114	0.318/0.318/1	0.318/0.318/1
	7	0.273/0.341/1.068	0.205/0.432/1.227	0.205/0.409/1.250	0.364/0.364/1	0.364/0.364/1
	9	0.205/0.364/1.159	0.182/0.409/1.227	0.182/0.409/1.227	0.409/0.409/1	0.364/0.364/1
	11	0.273/0.545/1.273	0.205/0.568/1.364	0.205/0.568/1.364	0.205/0.205/1	0.273/0.273/1
Vehicle	3	0.260/0.260/1	0.256/0.276/1.035	0.232/0.307/1.165	0.315/0.315/1	0.366/0.366/1
	5	0.299/0.299/1	0.280/0.319/1.067	0.264/0.362/1.150	0.417/0.417/1	0.472/0.472/1
	7	0.276/0.276/1	0.291/0.331/1.055	0.283/0.331/1.079	0.394/0.394/1	0.398/0.398/1
	9	0.354/0.354/1	0.339/0.390/1.063	0.299/0.402/1.146	0.402/0.402/1	0.472/0.472/1
	11	0.315/0.315/1	0.358/0.417/1.067	0.311/0.429/1.161	0.535/0.535/1	0.567/0.567/1
Soybean	3	0.117/0.170/1.085	0.120/0.189/1.106	0.082/0.215/1.247	0.314/0.314/1	0.303/0.303/1
	5	0.101/0.152/1.077	0.117/0.191/1.106	0.082/0.184/1.199	0.343/0.343/1	0.327/0.327/1
	7	0.088/0.202/1.149	0.109/0.253/1.189	0.072/0.234/1.295	0.327/0.327/1	0.237/0.237/1
	9	0.072/0.160/1.106	0.104/0.191/1.101	0.061/0.168/1.146	0.242/0.242/1	0.221/0.221/1
	11	0.088/0.213/1.157	0.088/0.229/1.184	0.090/0.226/1.181	0.215/0.215/1	0.160/0.160/1
Artificial Data	3	0.050/0.050/1	0.060/0.060/1.020	0.030/0.070/1.080	0.060/0.060/1	0.060/0.060/1
	5	0.060/0.060/1	0.050/0.060/1.030	0.060/0.070/1.040	0.060/0.060/1	0.060/0.060/1
	7	0.060/0.060/1	0.060/0.060/1	0.040/0.090/1.080	0.080/0.080/1	0.070/0.070/1
	9	0.070/0.070/1	0.070/0.080/1.010	0.060/0.100/1.050	0.100/0.100/1	0.090/0.090/1
	11	0.080/0.080/1	0.060/0.090/1.030	0.140/0.180/1.100	0.130/0.130/1	0.110/0.110/1

If we compare the ambiguous results (values e) generated by the fusion methods and those obtained with the use of the neural network, then it can be concluded that the results generated by the neural network are worse, in most cases not much (for example the Soybean data set). However, such a comparison is not objective, as this improvement is obtained by allowing ambiguity. It is only in the case of the artificial data set that the neural network approach produces better results. This is due to the fact that the artificial data have numerical attributes and the fusion methods do not generate as much ambiguity.

Table 4 presents the e_{ONE} values obtained with the use of neural network (best for all tested numbers of neurons in the hidden layer) and the considered fusion methods. The smallest ambiguity errors have been marked in bold.

Table 4. Classification ambiguity error e_{ONE} for neural network and other fusion methods.

Data Set	No. on Local Tables	Neural Network e_{ONE}	Sum Rule e_{ONE}	Borda Count e_{ONE}	Majority Vote e_{ONE}	Decision Templates e_{ONE}	Theory of Evidence e_{ONE}
Lypmho graphy	3	0.246	0.250	0.386	0.409	**0.159**	0.273
	5	**0.194**	0.318	0.318	0.318	0.318	0.318
	7	**0.308**	0.341	0.432	0.409	0.364	0.364
	9	**0.249**	0.364	0.409	0.409	0.409	0.364
	11	0.244	0.545	0.568	0.568	**0.205**	0.273
Vehicle	3	0.274	**0.260**	0.276	0.307	0.315	0.366
	5	**0.278**	0.299	0.319	0.362	0.417	0.472
	7	0.281	**0.276**	0.331	0.331	0.394	0.398
	9	**0.275**	0.354	0.390	0.402	0.402	0.472
	11	**0.274**	0.315	0.417	0.429	0.535	0.567
Soybean	3	**0.084**	0.170	0.189	0.215	0.314	0.303
	5	**0.085**	0.152	0.191	0.184	0.343	0.327
	7	**0.087**	0.202	0.253	0.234	0.327	0.237
	9	**0.073**	0.160	0.191	0.168	0.242	0.221
	11	**0.061**	0.213	0.229	0.226	0.215	0.160
Artificial Data	3	0.090	**0.050**	0.060	0.070	0.060	0.060
	5	**0.023**	0.060	0.060	0.070	0.060	0.060
	7	**0.010**	0.060	0.060	0.090	0.080	0.070
	9	**0.026**	0.070	0.080	0.100	0.100	0.090
	11	**0.036**	0.080	0.090	0.180	0.130	0.110
average e_{ONE}		0.160	0.227	0.262	0.274	0.269	0.275

Based on results from Table 4, it can be seen that the best results are obtained for the neural network approach. Moreover, it can also be seen that for more dispersed data (on a larger number of local tables) the proposed model still generates good results. On the other hand, other fusion methods generate increasingly worse results in the case of large dispersion of data.

The Friedman's test was performed. All results were selected—each fusion method as a separate group, the test confirmed that differences among the classification ambiguity error in these four groups are significant, with a level of $p < 0.000001$. Then, to determine the pairs of groups between which statistically significant differences occur, the Wilcoxon each pair test for dependent groups were performed. The test showed that there is significant difference with $p < 0.05$ for all pairs except between:

- Group 3 (Borda count) and two other groups (Group 5—Decision templates and Group 6—Theory of evidence),
- Group 4 (Majority voting) and two other groups (Group 5—Decision templates and Group 6—Theory of evidence),
- Group 5 (Decision templates) and Group 6 (Theory of evidence).

Additionally, comparative box-plot chart for the values of the classification ambiguity error was created (Figure 3). As can be observed, distributions of the classification ambigu-

ity error values in groups are completely different—especially better results were obtained for the proposed neural network method.

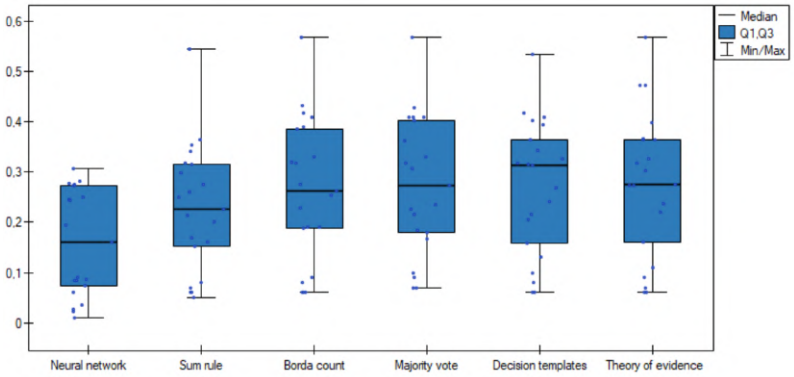


Figure 3. Box-plot chart with (Median, the first quartile—Q1, the third quartile—Q3) the value of classification ambiguity error e for the neural network and other fusion methods.

All experiments were performed on a portable computer with the following technical specifications:

- Intel i7-8565U CPU,
- 16 GB of RAM memory,
- Ubuntu 18.04.5 LTS operating system.

The algorithm has been implemented in Python and all the data-related calculations have been saved in a text document.

The main advantages and limitations of the proposed approach using the k-nearest neighbors algorithm and the neural network are listed below.

The main advantages are:

- The proposed model always generates unambiguous results for both numerical and qualitative data set.
- When the unambiguous results are compared, a much better quality of classification are obtained using the proposed model than using the other fusion methods: the majority voting, the Borda count method, the sum rule, the method that is based on decision templates and the method that is based on theory of evidence.
- The deviation of the results obtained by the proposed model is much smaller compared to the deviation of the results obtained for other fusion methods, which can be seen on Figure 3.
- Based on the performed experiments, it can be concluded that in most cases the number of neurons in the hidden layer equal to 4× the number of neurons in the input layer generates the best results. This parameter value can be adopted for related future work without the need to use a complex analysis of the network structure for each new data set separately.

The main limitations are:

- Unfortunately, neural networks do not provide a clear and human-interpretable formula for global decision making. A human readable principle, a rule for combining decisions or some pattern, is not generated.
- To apply the proposed model, we must have quite a large test set available to train the neural network. In the case of other fusion methods, such a condition does not have to be met. They can generate a global decision even in the case of only one test object being available.

4. Conclusions

In this article, a new model using a modified k-nearest neighbors algorithm in conjunction with a neural network to generate decisions based on dispersed data—available

from independent data sources is proposed. The paper presents a comparison of the results obtained with the use of the proposed model in comparison with other fusion methods known from the literature. This comparison was made using the data sets from the repository and an artificially generated data set. Both numerical and qualitative data were checked. Moreover, various degrees of dispersion into local tables of the analyzed data were analyzed. In the article, various structures of the neural network in terms of the number of neurons in the hidden layer were studied.

The obtained results show that the proposed model always generates unambiguous decisions. This is a big advantage of this approach as fusion methods very often generate ties, especially for qualitative data. In addition, it can be seen that the proposed model can deal equally well with data that is finely dispersed (11 local decision tables) and those that are less dispersed (three local decision tables). This statement is not true for other fusion methods which definitely produce worse results for highly dispersed data.

The comparison of the ambiguity classification error clearly showed that the proposed model generates better results than the other considered fusion methods. When accepting ties, fusion methods especially the Borda count method and the majority voting produce better results, but this is the result of allowing for ambiguity. Moreover, the results obtained with the use of the proposed approach are characterized by a much smaller deviation.

In the paper, the optimal number of neurons in the hidden layer of the neural network was determined. It is similar for all analyzed data sets; therefore, it can be used in related future works for new data sets.

The main limitations of the proposed approach are the need to provide a quite large test set to train the neural network. The model does not generate clear and interpretable rules according to which the global decision is determined.

In future research, it is planned to use different activation functions in the neural network. It is also planned to allow for ambiguity by applying multiple logistic regressions function in the output layer of neural network.

It would be very interesting to compare the performance of the proposed model with neural networks for different parameters of the data sets. It is planned to perform comparative experiments in future research:

- generating artificial data with a different number of conditional attributes, e.g., from 20 to 100 in steps of 20; the influence of the dimensionality of the data set;
- generating artificial data of different noise intensity; use the normal distribution with the standard deviation from 0.1 to 0.5 in steps of 0.1; the influence of the noise intensity;
- generating artificial data with a different number of outliers.

The presented studies focus on showing that the proposed approach has potential and will be developed in the future.

Author Contributions: Conceptualization, M.P.-K.; methodology, M.P.-K., K.F.M.; software, K.F.M.; validation, K.F.M.; formal analysis, M.P.-K., K.F.M.; investigation, M.P.-K., K.F.M.; writing—original draft preparation, M.P.-K.; writing—review and editing, M.P.-K., K.F.M.; visualization, M.P.-K., K.F.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available data sets were analyzed in this study. These data can be found here: [25]. One data set has been artificially generated description of the way is presented in the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Process. Mag.* **2020**, 37, 50–60. [CrossRef]
2. Thorgeirsson, A.T.; Gauterin, F. Probabilistic predictions with federated learning. *Entropy* **2021**, *23*, 41. [CrossRef] [PubMed]
3. Varghese, B.; Wang, N.; Nikolopoulos, D.S.; Buyya, R. Feasibility of fog computing. In *Handbook of Integration of Cloud Computing, Cyber Physical Systems and Internet of Things*; Springer: Cham, Switzerland, 2020; pp. 127–146.
4. Yang, Q.; Liu, Y.; Cheng, Y.; Kang, Y.; Chen, T.; Yu, H. Federated learning. *Synth. Lect. Artif. Intell. Mach. Learn.* **2019**, *13*, 1–207. [CrossRef]
5. Pfitzner, B.; Steckhan, N.; Arnrich, B. Federated Learning in a Medical Context: A Systematic Literature Review. *ACM Trans. Internet Technol. (TOIT)* **2021**, *21*, 1–31. [CrossRef]
6. Burduk, R.; Biedrzycki, J. Integration and selection of linear svm classifiers in geometric space. *JUCS J. Univers. Comput. Sci.* **2019**, *25*, 718.
7. Trajdos, P.; Burduk, R. Combination of linear classifiers using score function–analysis of possible combination strategies. In Proceedings of the International Conference on Computer Recognition Systems, Polanica-Zdrój, Poland, 20–22 May 2019; Springer: Cham, Switzerland, 2019; pp. 348–359.
8. Kuncheva, L.I. *Combining Pattern Classifiers: Methods and Algorithms*; John Wiley & Sons: Hoboken, NJ, USA, 2014.
9. Xiao, P.; Cheng, S.; Stankovic, V.; Vukobratovic, D. Averaging is probably not the optimum way of aggregating parameters in federated learning. *Entropy* **2020**, *22*, 314. [CrossRef] [PubMed]
10. Zhang, X.Q.; Wang, H.B.; Yu, H.Z. Neural network based algorithm and simulation of information fusion in the coal mine. *J. China Univ. Min. Technol.* **2007**, *17*, 595–598. [CrossRef]
11. Wang, J.; Gao, Y.; Liu, W.; Sangaiah, A.K.; Kim, H.J. An intelligent data gathering schema with data fusion supported for mobile sink in wireless sensor networks. *Int. J. Distrib. Sens. Netw.* **2019**, *15*, 1550147719839581. [CrossRef]
12. Przybyła-Kasperek, M.; Wakulicz-Deja, A. Global decision-making system with dynamically generated clusters. *Inform. Sci.* **2014**, *270*, 172–191. [CrossRef]
13. Kuncheva, L.I.; Whitaker, C.J. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Mach. Learn.* **2003**, *51*, 181–207. [CrossRef]
14. Przybyła-Kasperek, M. Ensemble of Classifiers Based on Genetic Reducts and K-Nearest Neighbors Classifier for Data with Non Missing Values. In Proceedings of the Information Systems Development: Crossing Boundaries between Development and Operations (DevOps) in Information Systems (ISD2021 Proceedings), Valencia, Spain, 8–10 September 2021.
15. Przybyła-Kasperek, M. Three conflict methods in multiple classifiers that use dispersed knowledge. *Int. J. Inf. Technol. Decis. Mak.* **2019**, *18*, 555–599. [CrossRef]
16. Glorot, X.; Bordes, A.; Bengio, Y. Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, Ft. Lauderdale, FL, USA, 14 June 2011; pp. 315–323.
17. Li, X.; Li, X.; Pan, D.; Zhu, D. On the learning property of logistic and softmax losses for deep neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–22 February 2020; Volume 34, pp. 4739–4746.
18. Bishop, C.M. Pattern Recognition and Machine Learning. In *Information Science and Statistics*; Springer: Berlin, Germany, 2006.
19. Kingma, D.P.; Ba, J.L. Adam: A Method for stochastic Optimization. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
20. Mannor, S.; Peleg, D.; Rubinstein, R. The cross entropy method for classification. In Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 7–11 August 2005; pp. 561–568.
21. Kuncheva, L.; Bezdek, J.C.; Duin, R.P.W. Decision templates for multiple classifier fusion: An experimental comparison. *Pattern Recognit.* **2001**, *34*, 299–314. [CrossRef]
22. Przybyła-Kasperek, M.; Wakulicz-Deja, A. Dispersed decision-making system with fusion methods from the rank level and the measurement level—A comparative study. *Inf. Syst.* **2017**, *69*, 124–154. [CrossRef]
23. Rogova, G.L. Combining the results of several neural network classifiers. *Neural Netw.* **1994**, *7*, 777–781. [CrossRef]
24. Russell, I.; Markov, Z. An introduction to the Weka data mining system. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, Seattle, WA, USA, 8–11 March 2017; p. 742.
25. Asuncion, A.; Newman, D.J. *UCI Machine Learning Repository*; University of Massachusetts Amherst: Amherst, MA, USA, 2007. Available online: https://archive.ics.uci.edu (accessed on 25 October 2021).

Publication [P2]

Przybyła-Kasperek M., Marfo K.F. Influence of noise and data characteristics on classification quality of dispersed data using neural networks on the fusion of predictions *International Conference Information Systems Development*, 2022. doi: 10.62036/ISD.2022
URL: http://dx.doi.org/10.62036/ISD.2022.
DOI: 10.62036/ISD.2022.
$\text{MEiN}_{2022} = 140$
Number of citations:

- according to Web of Science: 0
- according to Google Scholar: 3

Contributor	Description of main tasks
Małgorzata Przybyła-Kasperek	- conceptualization and methodology - result analysis - manuscript: original draft preparation - manuscript: review and editing - visualization: creating all figures in manuscript
Kwabena Frimpong Marfo	- investigation - conceptualization and methodology - result analysis - implementation of proposed algorithm - manuscript: review and editing - visualization: creating all figures in manuscript

Influence of Noise and Data Characteristics on Classification Quality of Dispersed Data Using Neural Networks on the Fusion of Predictions

Małgorzata Przybyła-Kasperek
University of Silesia in Katowice
Katowice, Poland *malgorzata.przybyla-kasperek@us.edu.pl*

Kwabena Frimpong Marfo
University of Silesia in Katowice
Katowice, Poland *kwabena.marfo@us.edu.pl*

Abstract

In this paper, the issues of classification based on dispersed data are considered. For this purpose, an approach is used in which prediction vectors are generated locally using the k-nearest neighbors classifier. However, in central server, the final fusion of prediction vectors is made with the use of a neural network. The main aim of the study is to check the influence of various data characteristics (the number of conditional attributes, the number of objects, the number of decision classes) and the degree of dispersion and noise intensity on the quality of classification of the considered approach. For this purpose, 270 data sets were generated that differed by the above factors. Experiments were carried out using these data sets and statistical tests were performed. It was found that each of the examined factors has a statistically significant impact on the quality of classification. However, the number of conditional attributes, degree of dispersion, and noise intensity have the greatest impact. Multidimensionality in dispersed data affects the results positively, but the analyzed method is only resistant to a certain degree of noise intensity and dispersion.

Keywords: Federated Learning, Dispersed Data, Neural Network, Noise, Degree of Dispersion.

1. Introduction

Many different classification methods have been proposed in machine learning so far. These methods have found applications in numerous information systems used in banking, stock exchange, electronic markets, medicine, among others. Most of the traditional classification methods are dedicated to data stored in one decision table, yet, this approach is increasingly seen as insufficient. In today's global society, federated learning is a critical issue [17]. This approach responds to the widespread occurrence of dispersed data provided by various units that wish to keep their data private. In all of the applications mentioned above, examples of dispersed data can be found [10]. Currently, we are dealing with data collected in a dispersed manner by various units, institutions, websites, and mobile devices [1]. When local data is used together, better quality of classification can be obtained than when we rely on one fragment of data. Even so, using dispersed data is not a simple task. First and foremost, there is a great possibility of the presence of inconsistencies in the data – independently collected data may have different set of attributes as well as different set of objects, but the possibility of having common elements among dispersed data is not excluded. In this case, it is not possible to merge such data into one table. Secondly, the difficulty of using dispersed data stems from the fear of freely sharing data. Often, data-owners want to preserve data privacy, thus, we cannot construct a method that accesses all data from various sources.

Scientists wonder if it is necessary to propose many methods for solving real problems [4].

On the other hand, we have a no-free lunch theorem [1] which justifies proposing new methods dedicated to specific problems. In the context of such considerations, it is important to characterize methods in terms of the problem and data characteristics for which the method is dedicated to. To do this, the method should be tested in terms of both varying data and different contexts. The differences in data can be considered in terms of the following characteristics: the number of objects, the number of conditional attributes, the number of decision classes, informativeness and redundancy of attributes. Other important concepts to consider are presence of imbalance decision classes and the presence of noise in the data.

As an instance, suppose we want to build an automated diagnostic system using records of patients across multiple hospitals. Depending on the above mentioned data characteristics, the performance of the learning algorithm could be greatly affected and one would not have a clear approach to address this issue. However, with the use of an information system, we could drill down to the factors that truly cause the deterioration in the learning algorithm. In the paper [12], a method for classification with the use of dispersed data was proposed. This method uses the k-nearest neighbors and the neural networks classifiers. The k-nearest neighbors classifier is used for local data – a prediction vector is generated, which is then transferred to a central server. Then the neural network makes the final decision based on all prediction vectors. In this way, we maintain data privacy as only the prediction vectors are shared. The paper [12] presents that this approach gives unambiguous results, which cannot be said about other fusion methods such as the Majority Voting, the Borda Count method, the Sum Rule, the method based on decision templates and the method based on theory of evidence. Moreover, it was shown that the approach with neural networks achieves better quality of classification than the above-mentioned fusion methods.

However, there are still many questions to be answered. Does the number of conditional attributes/the number of objects/the number of decision classes in local data affect the quality of classification? Does information noise disturb the classification of dispersed data with the use of neural networks, and if so, to what extent? If the informative attributes are dispersed among local data, are we able to make a good classification based on fragmented data? How does high data dispersion affect the quality of classification? The aim of this paper is to answer these questions. In this way, the scope of applications of the classification method for dispersed data using the k-nearest neighbors and the neural network classifiers will be determined.

The article focuses on the extent to which data characteristics mentioned above, as well as noise in data affects the classification accuracy of method proposed in this paper. This issue is very important in the case of dispersed data with noise, for example, data from social media. For this purpose, 18 data sets were generated with varying number of objects, attributes and decision classes. To each of them, noise was added in three different degrees of intensity. The data was divided into five different versions of dispersion (with different number of local data) in such a way as to guarantee different, but not necessarily disjoint sets of attributes in each of the local sets. Thus, 270 dispersed data was obtained. Experiments using these data and different number of neurons in the hidden layer were performed. The obtained results were compared and conclusions were drawn.

The article is organized in the following way. Section 2 provides a literature overview. The next section describes the classification approach of the dispersed data and the method of data generation. In Section 4, the results of the experiments are presented and analyzed. The article ends with conclusion.

2. Literature Review

The issues of data stored in local sets are considered mainly in two contexts; an ensemble of classifiers [2, 3, 11] – where local data is created based on a single data set in order to improve the quality of classification. In this approach we have control over the form of local data created.

Local data meets certain conditions, for example, independence and variety. In this paper, a completely different approach is considered because dispersed data do not have to fulfill any of these constraints.

Dispersed data is also considered in federated learning topics [5, 14]. Similar to the data considered in this paper, the local data are collected independently and we have no influence on their form. Federated learning involves applying machine learning methods locally to each local device separately, without sharing the training objects with a central server [10]. This approach makes it possible to learn a common model while maintaining data privacy. Federated learning issues are widely used in applications such as smart healthcare, smart transportation, Unmanned Aerial Vehicles, smart cities, and smart industry [9].

Neural networks in the context of federated learning are considered in many papers. In [16], neural networks are built locally, and their weights are sent to a central server in order to build the final model. The paper [6] also analyzes the use of neural networks in the context of federated learning, but the main focus is on passing the weights to the central server more efficiently. In [8] a model with a deep graph neural network is proposed to classify the nodes based on their structures and features. In this paper, however, the combination of the k-nearest neighbors classifier, which will be used locally, with a neural network, which will be built on a central server, is considered. Such an approach was investigated in [12] although the influence of various data characteristics, degree of dispersion and noise intensity on the quality of classification of this method was not analyzed there. Such studies are performed in this work.

3. Methods and Data

In this section, we first briefly introduce the dispersed data classification approach that uses the k-nearest neighbors and the neural network classifiers. Next, the method of generating and preprocessing data sets used in the experimental analysis is described.

3.1. Dispersed Classification Method with k-Nearest Neighbors and the Neural Network Classifiers

The approach used in this paper to classify based on dispersed data consists of two steps. We assume that each local data is stored in the form of a local decision table. In the first stage, the calculations are performed independently in local destinations. A modified k-nearest neighbors algorithm is used and predictions from the measurement level are designated. We assume that a set of decision tables $D_{ag} = (U_{ag}, A_{ag}, d)$, $ag \in Ag$ from one discipline is available, where U_{ag} is the universe, a set of objects; A_{ag} is a set of conditional attributes; d is a decision attribute. Based on each of the local tables, a classifier is built. Ag is a set of classifiers, and ag is a single classifier. For each local table and for each test object x, a probability vector over decision classes (denoted by $\mu_{ag}(x)$) is designated. The dimension of vectors $\mu_{ag}(x) = [\mu_{ag,1}(x), \ldots, \mu_{ag,c}(x)]$ is equal to the number of decision classes $c = card\{V^d\}$, where V^d is a set of values of decision attributes from all decision tables and $card\{V^d\}$ is the cardinality of this set. Each coefficient $\mu_{ag,j}(x)$ is determined using the k-nearest neighbors of the test object x belonging to a given decision class j and decision table D_{ag}. The gower similarity measure is used in this approach. For numerical data used in this paper, the gower measure is equivalent to the Manhattan distance.

Only the prediction vectors are made available for centralized computation. A global decision for a test object is generated with the use of a neural network. The structure of the network consists of three layers. The hidden layer has a varying number of neurons that will be studied experimentally. The number of neurons in the input layer is equal to the product of the number of prediction vectors and the dimension of the vector, i.e. $card\{Ag\} \times card\{V^d\}$. The output layer has the number of neurons equal to the number of decision classes. For the hidden layer,

the ReLU (Rectified Linear Unit) activation function is used. For the output layer, the SoftMax activation function is used, which is recommended when we deal with a multi-class problem [7]. The back-propagation method, the Adam optimizer and the categorical cross-entropy loss function are used in the study.

Of course, a neural network must be trained using a certain set of objects. Since the training objects were used to generate the prediction vectors with the k-nearest neighbors classifier, they cannot be reused to train the neural network. Thus, a 10-fold cross-validation method was used for the test set. Each time, the neural network was trained using 9 folds, while the last independent fold was classified using the neural network constructed based on the 9 folds. This procedure was repeated ten times for each of the folds, and the final quality of classification was assigned using the results obtained from these ten performances. For a more detailed description, please refer to [12].

The big advantage of the approach described above is that it generates unambiguous decisions, which cannot be said about many other fusion methods [1]. Moreover, in comparison with fusion methods such as the Majority Voting, the Borda Count method, the Sum Rule, the method based on decision templates and the method based on theory of evidence; the approach described above gives in most cases a better quality of classification. But it is obvious that the proposed approach is not appropriate in every case or for every data set. Therefore, the question remains – with which data does the dispersed classification method with neural network handles best. To answer this question, data sets were generated that differ in many factors. A total of 270 dispersed data sets were tested, which is described below.

3.2. Data

The data was generated artificially as the aim was to systematically compare the results obtained from data with specific characteristics. The aim was to compare results obtained with the use of dispersed classification method with neural network in relation to the following issues with respect to the impact on the performance of the proposed algorithm:

- the impact that the number of conditional attributes in data has on the quality of classification (multidimensionality of data),
- the impact that the number of objects in data has on the quality of classification,
- the impact that the number of decision classes in data has on the quality of classification,
- the impact that the degree of dispersion has on the quality of classification,
- the impact that the noise intensity in data has on the quality of classification.

The generation of artificial data sets was carried out in several stages:

1. In the first stage, data sets were generated using the Weka [13] software. For this purpose, the RandomRBF was used. This function, at first, randomly generates centers for each decision class. Then to each center, a weight is randomly assigned as well as a central point per attribute, and a standard deviation. A new object is generated as follows; a center is selected according to the weights. Then attribute values are randomly generated and offset from the center. After, the vector is scaled so that its length is equal to a value sampled randomly from the Gaussian distribution of the center. In this way, 18 data sets were generated with different number of conditional attributes, decision classes, objects and centroids. The number of objects in decision classes is imbalanced. The characteristics are presented in Table 1.

There are two main reasons why the training and testing methods used in this paper are the best methods to evaluate the quality of classification for dispersed data. To begin, when

Table 1. Data set characteristics.

Data Set	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
No. of objects	650	650	650	1300	1300	1300	650	650	650	1300	1300	1300	650	650	650	1300	1300	1300
No. of conditional attributes	30	50	70	30	50	70	30	50	70	30	50	70	30	50	70	30	50	70
No. of decision classes	10	10	10	10	10	10	5	5	5	5	5	5	5	5	5	5	5	5
No. of centroids	100	100	100	100	100	100	50	50	50	50	50	50	100	100	100	100	100	100

we deal with dispersed data, it significantly increases computational complexity. Also, there are different sets of conditional attributes in different local tables, and the test object must have specific values on all of these attributes. Thus, at first, each of the data sets was randomly but in a stratified way divided into training set (70% of the data) and testing set (30% of the data).

One of the study goals was to check the influence of noise intensity on the quality of classification. For this purpose, three different levels of noise intensity were applied to the training set (70% of the data), thus, the density of the noise was 100% of the training set. After dividing each data into training and testing set, 3 training data sets with Gaussian noise intensity were further constructed. For each set, mean value equal to 0 and different values of standard deviation (std) were used: $\{mean = 0, std = 0.01\}, \{mean = 0, std = 0.1\}$ and $\{mean = 0, std = 0.2\}$ respectively. In this way, based on 18 training sets, 54 data sets with different noise intensities were created.

Another research goal was to check the influence of the degree of dispersion on the quality of classification. Each of the 54 data sets prepared in the previous step was divided into five versions of dispersion – $3, 5, 7, 9, 11$ local tables were constructed from each training data set with different number of conditional attributes. Local decision tables were constructed in a way such that each local table has a unique set of conditional attributes and also some conditional attributes that are present in other local tables. The number of attributes in local tables varied from 3 to 35. For a finer dispersion, a greater number of local tables contain a smaller number of conditional attributes.

The above described approach is used to generate 270 dispersed data. Thus, for each of the original training set (70% of the data), we have dispersed data with 3, 5, 7, 9, 11 local tables for $mean = 0, std \in \{0.01, 0.1, 0.2\}$ Gaussian noise intensities.

The code of the function that defines the Gaussian noise intensities for each training data set is given in Listing 1.

Listing 1. Gaussian Noise Intensity for Training Data Sets

```
# df: Original training data set read into a Dataframe
# col: A list of conditional attributes
# mu: mean value for Gaussian noise
# sigma: standard deviation value for Gaussian noise

def get_noise(df, col:list, mu:float,sigma:float):
noise = np.random.normal(mu, sigma, size(df))
return pandas.DataFrame(noise, columns=col)
```

The quality of classification was evaluated based on the test set using the estimator of classification error e. It is defined as a fraction of the total number of objects in the test set that were classified incorrectly. With the use of the analyzed approach, the decisions generated are always unambiguous – thus, one decision class is always generated by the system.

Table 2. Results of classification error e for the dispersed system with neural network

	Noise $std = 0.01$					Noise $std = 0.1$					Noise $std = 0.2$				
Data	No. of local tables														
Set	3	5	7	9	11	3	5	7	9	11	3	5	7	9	11
1	0.039	0.049	0.059	0.09	0.138	0.049	0.065	0.085	0.133	0.247	0.179	0.241	0.324	0.483	0.636
2	0.029	0.034	0.026	0.032	0.031	0.026	0.029	0.013	0.036	0.031	0.041	0.056	0.076	0.092	0.142
3	0.003	0	0	0.001	0	0.005	0	0	0.001	0.003	0.006	0.013	0.022	0.016	0.016
4	0.017	0.035	0.04	0.058	0.094	0.028	0.065	0.058	0.088	0.203	0.069	0.138	0.383	0.584	0.699
5	0.001	0.002	0.003	0.005	0.006	0.001	0.003	0.005	0.006	0.012	0.002	0.018	0.036	0.084	0.152
6	0	0	0	0	0	0	0	0	0	0	0	0.002	0.006	0.005	0.035
7	0.017	0.022	0.029	0.029	0.041	0.01	0.025	0.03	0.035	0.071	0.018	0.041	0.063	0.102	0.193
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0.003	0.001	0	0	0.004	0.002	0.008	0.007	0.005	0.01	0.007	0.007
10	0.01	0.017	0.019	0.028	0.031	0.01	0.021	0.017	0.029	0.042	0.017	0.024	0.035	0.072	0.123
11	0.002	0.002	0.001	0.002	0.005	0.002	0.002	0.002	0.002	0.005	0.002	0.004	0.003	0.003	0.005
12	0.001	0.003	0.002	0.003	0.003	0.003	0	0.002	0.003	0.003	0.002	0.003	0.003	0.003	0.003
13	0.048	0.058	0.068	0.077	0.115	0.056	0.058	0.061	0.09	0.145	0.073	0.131	0.211	0.366	0.591
14	0.005	0.006	0.011	0.008	0.013	0.01	0.008	0.015	0.01	0.016	0.008	0.025	0.027	0.041	0.051
15	0	0	0.005	0	0.003	0	0	0	0	0	0.003	0.008	0.011	0.005	0.023
16	0.007	0.017	0.026	0.047	0.064	0.016	0.023	0.031	0.055	0.078	0.024	0.047	0.11	0.347	0.636
17	0	0	0	0.001	0.001	0	0	0	0.001	0.003	0	0.001	0.01	0.021	0.028
18	0	0	0	0	0.005	0.001	0.001	0	0.001	0.003	0.002	0	0.002	0.003	0.01

4. Results

For each dispersed set (one of the 270 analyzed) the experiments were carried out according to the following scheme:

- Generating vectors of predictions based on local tables using the k-nearest neighbors classifier. For each data set, three different values of the k parameter were tested, namely $k \in \{1, 5, 10\}$. One parameter value was selected for each dispersed data set that produced the best overall results. For the majority of data – $k = 1$ were selected. Only for data set 4 and the dispersion with 9 and 11 local tables, $k = 5$ was selected.

- Generating a global decision using a neural network with one hidden layer and different number of neurons in the hidden layer. For each data set, the following number of neurons in the hidden layer were tested: $\{1, 3, 4, 4.25, 4.5, 4.75, 5\}\times$the number of neurons in the input layer. Different number of neurons in the hidden layer was also checked. However, it was noticed that the accuracy of the respective models improves as the number of neurons in the hidden layer increases, but significant improvement declines around $5\times$ the number of neurons in the input layer. The number of neurons in the input layer depends on the number of local tables. Thus, the more dispersed data we have, the more complex the structure of the neural network is.

It should be noted once again that to use the neural network, a 10-fold cross-validation was used on the test set, i.e., the neural network was trained 10 times with 9 folds and tested on one remaining fold. In addition, each test was performed three times to ensure that the results were reliable and not distorted by the influence of randomness. The results for the neural network approach that are given below are the average of the obtained results.

The results obtained for the optimal number of neurons in the hidden layer are presented in Table 2. We do not present the individual results obtained for a different number of neurons in the hidden layer ($\{1, 3, 4, 4.25, 4.5, 4.75, 5\}\times$ the number of neurons in the input layer) due to the limited space. However, for many data sets, the optimal number was around $4\times$ the number of neurons in the input layer. In Table 2, 18 data sets are listed in the rows, the columns distinguish between three different noise levels and five different versions of dispersion.

As can be seen, some data sets were trivial for the analyzed approach. These are data sets 6, 8, 9, 15, 17 and 18 for which the classification error was almost always equal to 0 regardless of the version of dispersion and noise intensity. As can be seen from data characteristics – Table 1, the main factor affecting data simplicity is the number of conditional attributes occurring in the data. The data sets 6, 9, 15 and 18 have 70 attributes (which were split into local tables). However, the approach of using dispersed data perfectly copes with the information stored in

local tables and makes correct decisions. For the two remaining data sets 8 and 17, the number of conditional attributes was equal to 50 - so it is also a large number, while here the factor influencing the simplicity of the data was a small number of centroids (for set 8) and a large number of objects (for set 17).

The general hypotheses that can be made based on the results in Table 2 in relation to the effect each data set had on the performance of the algorithm proposed in this paper are as follows:

- The more conditional attributes occurred in the data, the better the quality of classification. Multidimensionality is beneficial for dispersed data.
- The greater the number of training objects, the better the quality of classification.
- The greater the number of decision classes, the worse the quality of classification.
- For greater dispersion (number of local tables 3, 5, 7, 9, 11) the quality of classification deteriorates.
- The analyzed method is not immune to noise; high noise and significant dispersion (a large number of local tables) gives poorer quality of classification.

The tests of statistical significance for all of the above hypotheses are presented below.

4.1. Comparison of Experimental Results for Different Numbers of Conditional Attributes

In order to investigate how the number of conditional attributes affects the quality of classification for dispersed data and the approach using neural network, all results from Table 2 were used – each number of conditional attributes as a separate group. Thus, we have three independent samples for data with 30 conditional attributes, 50 conditional attributes, 70 conditional attributes. Each sample containing 90 observations – results obtained for different versions of dispersion and noise intensity with a constant number of conditional attributes. The Kruskal-Wallis test confirmed that differences among the classification error in these three groups are significant, with a level of $p = 0.000001$, $\chi^2(2) = 151.257$. Then, to determine the pairs of groups between which statistically significant differences occur, the Mann-Whitney test were performed. The test showed that there is a significant difference with $p < 0.0005$ between each pair.

Additionally, comparative box-plot chart for the values of the classification error was created (Figure 1) – the classification error values obtained for the data with different numbers of conditional attributes are presented. As can be observed, distributions of the classification error values in groups are very different. For dispersed data, multidimensionality is very good. The total number of conditional attributes occurring in local tables above 30 already gives a very good quality of classification when using neural network as a fusion method. Anyway, it also reflects the real situation when various units participate in making joint decision. The variety of attributes occurring in local tables has a positive effect on the quality of decisions made.

4.2. Comparison of Experimental Results for Different Numbers of Training Objects

In order to investigate how the numbers of training objects affects the quality of classification for dispersed data and the approach using neural network, all results from Table 2 were used – each number of training objects in data as a separate group. Thus, we have two independent samples for data with 455 training objects and 910 training objects. Each sample containing 135 observations – results obtained for different versions of dispersion and noise intensity with a constant number of training objects. The Mann-Whitney test for independent groups were performed. The test showed that there is significant difference with $p < 0.03$ between groups.

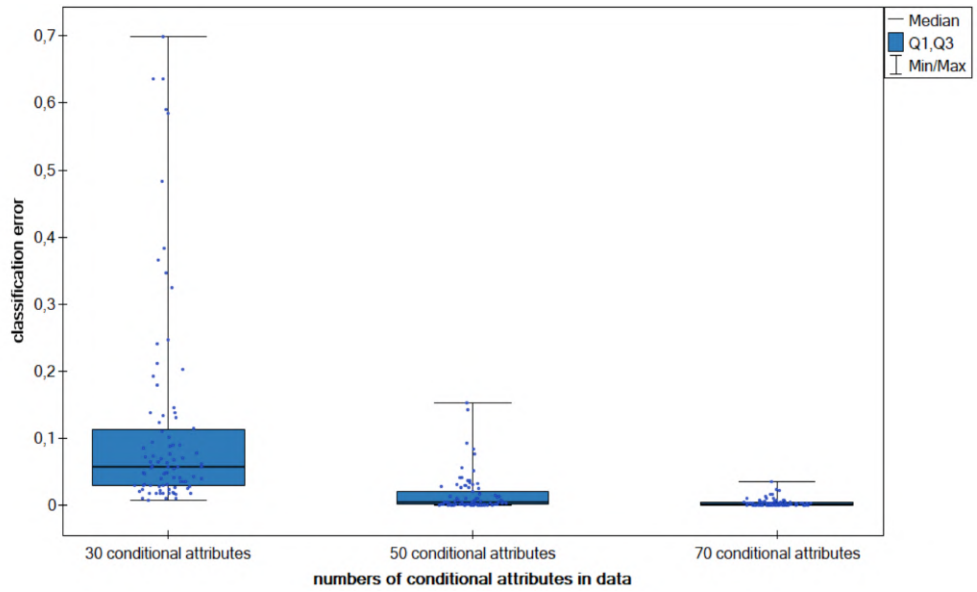


Fig. 1. Box-plot chart with (Median, the first quartile – Q1, the third quartile – Q3) the value of classification error e for the neural network with different numbers of conditional attributes in data.

Additionally, comparative box-plot chart for the values of the classification error was created (Figure 2) – the classification error values obtained for the data with different numbers of training object are presented. As can be observed, distributions of the classification error values in groups are not so different as at the previous graph. This means that the number of objects in dispersed data does not affect the quality of classification as much as the number of conditional attributes. Of course, the difference in results is statistically significant, so the more training objects, the better, but the total number of conditional attributes in dispersed data is much more important.

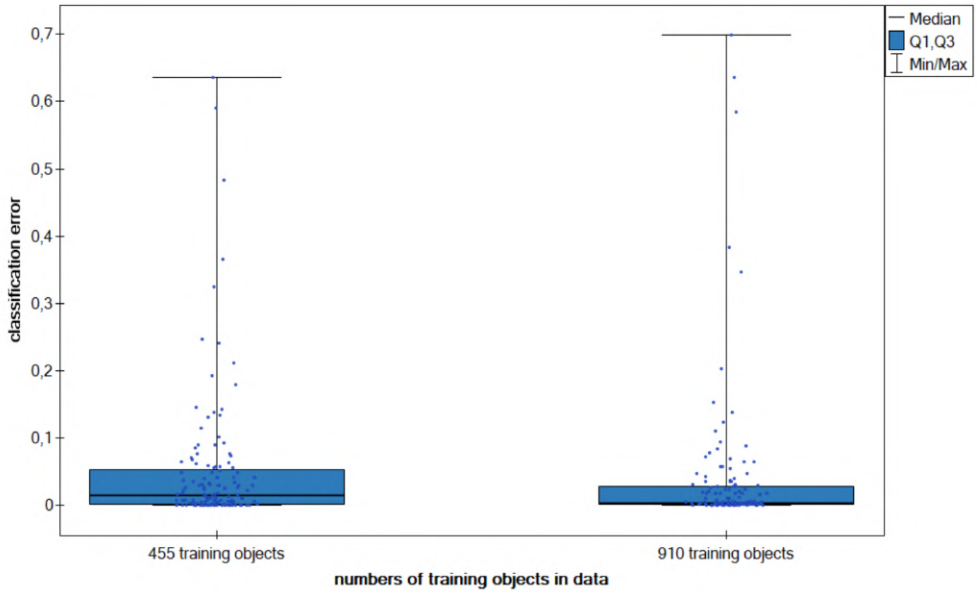


Fig. 2. Box-plot chart with (Median, the first quartile – Q1, the third quartile – Q3) the value of classification error e for the neural network with different numbers of training objects in data.

4.3. Comparison of Experimental Results for Different Numbers of Decision Classes

In order to investigate how the numbers of decision classes affects the quality of classification for dispersed data and the approach using neural network, all results from Table 2 were used – each number of decision classes in data as a separate group. Thus, we have two independent samples for data with 5 decision classes and 10 decision classes. The first sample contains 180 observations and the second sample contains 90 observations – results obtained for different versions of dispersion and noise intensity with a constant number of decision classes. The Mann-Whitney test for independent groups confirmed that there is significant difference with $p < 0.0005$ between groups.

Additionally, comparative box-plot chart for the values of the classification error was created (Figure 3) – the classification error values obtained for the data with different numbers of decision classes are presented. As can be observed, the difference between distributions of the classification error values in groups is less noticeable than for the number of conditional attributes, but more visible than for the number of training objects. It is more or less obvious that the more decision classes we have, the more difficult it is to make a correct decision. However, it can be concluded that the number of decision classes in the dispersed data set is less significant in terms of data difficulty than the number of conditional attributes.

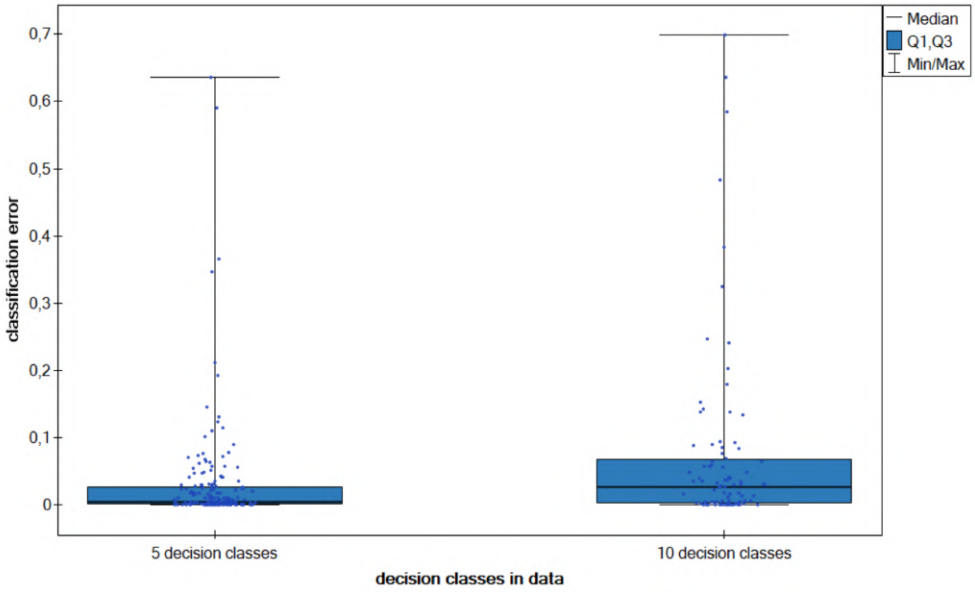


Fig. 3. Box-plot chart with (Median, the first quartile – Q1, the third quartile – Q3) the value of classification error e for the neural network with different number of decision classes in data.

4.4. Comparison of Experimental Results for Different Degree of Dispersion

As the degree of dispersion, we understand the number of local tables occurring in the dispersed data. In order to investigate how the numbers of local tables affects the quality of classification for dispersed data and the approach using neural network, all results from Table 2 were used – each number of local tables in data as a separate group. Thus, we have five dependent samples (as one data set was divided into a different number of local tables) for data with 3, 5, 7, 9 and 11 local tables. Each sample contain 55 observations – results obtained for different data sets and noise intensity with a constant number of local tables. The Friedman's test confirmed that differences among the classification error in these five groups are significant, with a level of $p = 0.000001$. Then, to determine the pairs of groups between which statistically significant

differences occur, the Wilcoxon pair test for dependent groups were performed. The test showed that there is significant difference with $p < 0.00004$ between each pair.

Additionally, comparative box-plot chart for the values of the classification error was created (Figure 4) – the classification error values obtained for the dispersed data with different numbers of local tables are presented. As can be observed, the difference between distributions of the classification error values in groups is most noticeable between the extreme degrees of dispersion (3 local tables and 11 local tables). The results can be summarized that the greater the dispersion, the worse the quality of classification.

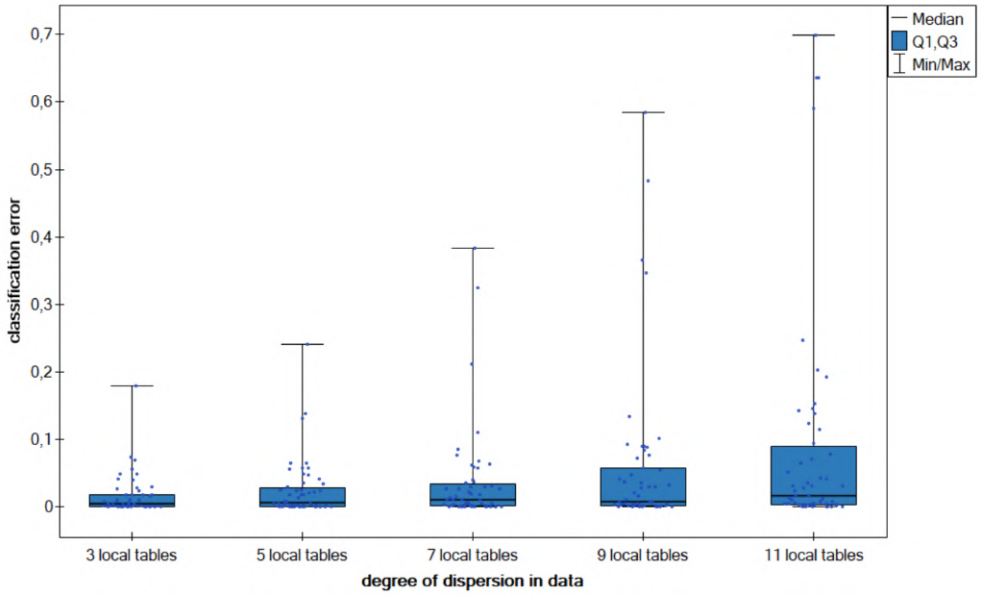


Fig. 4. Box-plot chart with (Median, the first quartile – Q1, the third quartile – Q3) the value of classification error e for the neural network with different degree of dispersion in data.

4.5. Comparison of Experimental Results for Different Noise Intensity in Data

In order to investigate how the intensity of noise affects the quality of classification for dispersed data and the approach using neural network all results from Table 2 were used – each noise intensity (Gaussian noise level with $std \in \{0.01, 0.1, 0.2\}$) in data as a separate group. Thus, we have three dependent samples (as three different noise levels were generated based on one data set) for data with noise $std = 0.01$, $std = 0.1$ and $std = 0.02$. Each sample contain 90 observations – results obtained for different data sets and number of local tables with a constant noise intensity. The Friedman's test confirmed that differences among the classification error in these three groups are significant, with a level of $p = 0.000001$. Then, to determine the pairs of groups between which statistically significant differences occur, the Wilcoxon pair test for dependent groups were performed. The test showed that there is significant difference with $p < 0.00002$ between each pair.

Additionally, comparative box-plot chart for the values of the classification error was created (Figure 5) – the classification error values obtained for the data with different noise intensity $std \in \{0.01, 0.1, 0.2\}$ are presented. For noise intensities equal to $std = 0.01$ and $std = 0.1$, the difference in the distributions is not so noticeable. Only when the noise intensity significantly increased to the level of $std = 0.2$, we can observe a significant increase in the classification error in comparison with the lower noise level. This means that although the classification method for dispersed data with neural network is immune to noise to some extent, it does not cope well with information noise at the $std = 0.2$ level.

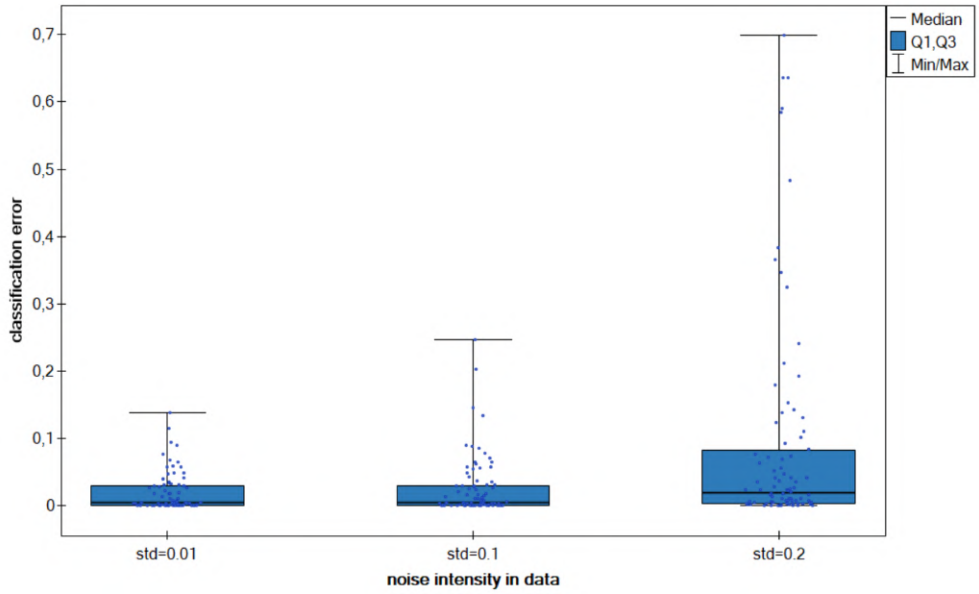


Fig. 5. Box-plot chart with (Median, the first quartile – Q1, the third quartile – Q3) the value of classification error e for the neural network with different noise intensity in data.

5. Conclusion

In the paper, the approach to classification based on dispersed data was analyzed, in which the k-nearest neighbors algorithm was used as a local classifier and the neural network as a fusion method. The analyzed approach takes into account the assumptions of data protection and privacy like in federated learning domain and does not interfere with the form of local data. The main aim of the study was to examine the impact of different data characteristics, the degree of dispersion and noise intensity on the classification quality of the above-mentioned approach. For this purpose, 270 different data sets were generated and experiments were performed. Analysis of obtained results and statistical tests were made.

The main conclusions are as follows. Regarding the data sets characteristics, in the case of dispersed data, the number of conditional attributes has the greatest impact on the quality of classification. Multidimensional dispersed data guarantees better quality of classification. The second most important factor is the number of decision classes. The fewer decision classes, the easier the set for classification. The number of training objects has the least influence on the quality of classification. The more objects in the data set, the better the quality of classification we get. The degree of dispersion has a significant impact on the quality of classification. With significant dispersion (11 local tables), the quality of classification significantly decreases in comparison with the results obtained for data with low dispersion (3 local tables). In the case of small dispersion (3, 5, 7 local tables) the impact on the results is not that significant. Noise occurrence also negatively affects the quality of classification. To some extent it can be said that the method is immune to noise. For Gaussian noise level with an average value 0 and standard deviation not exceeding 0.1, the method generates results in which the difference is not that drastic. Nevertheless, for the Gaussian noise level with the standard deviation equal to 0.2, large decrease in the quality of classification has been noted. The method does not cope well with a high noise level.

In future study, it is planned to propose a classification method for dispersed data, which enables conflict analysis and coalitions creation in order to eliminate the negative impact of high degree of dispersion on the quality of classification. In addition, it is planned to apply the proposed approach to dispersed business data, more specifically stock exchange data.

References

1. Adam, S. P., Alexandropoulos, S. A. N., Pardalos, P. M., Vrahatis, M. N.: No free lunch theorem: A review. Approximation and optimization, 57–82, (2019)
2. Blachnik, M.: Ensembles of instance selection methods: A comparative study. International Journal of Applied Mathematics and Computer Science, 29(1), (2019)
3. Bolon-Canedo, V., Alonso-Betanzos, A.: Ensembles for feature selection: A review and future trends. Information Fusion, 52, 1–12, (2019)
4. Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D.: Do we need hundreds of classifiers to solve real world classification problems? The journal of machine learning research, 15(1), 3133–3181, (2014)
5. Kołodziej, T., Rościszewski, P.: Towards Scalable Simulation of Federated Learning. In International Conference on Neural Information Processing, 248–256. Springer, Cham, (2021)
6. Konecny, J. H., McMahan, B., Yu, X., Richtarik, P., Suresh, A.T., Bacon, D.: Federated Learning: Strategies for Improving Communication Efficiency, NIPS Workshop on Private Multi-Party Machine Learning (2016)
7. Li, X.; Li, X.; Pan, D.; Zhu, D. On the learning property of logistic and softmax losses for deep neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, 34, 4739–4746, (2020)
8. Mei, G., Guo, Z., Liu, S., Pan, L.: SGNN: A Graph Neural Network Based Federated Learning Approach by Hiding Structure, 2019 IEEE International Conference on Big Data (Big Data), 2560–2568, (2019)
9. Nguyen, D. C., Ding, M., Pathirana, P. N., Seneviratne, A., Li, J., Poor, H. V.: Federated learning for internet of things: A comprehensive survey. IEEE Communications Surveys and Tutorials, (2021)
10. Pfitzner, B., Steckhan, N., Arnrich, B.: Federated learning in a medical context: A systematic literature review. ACM Transactions on Internet Technology (TOIT), 21(2), 1–31, (2021)
11. Pławiak, P.: Novel genetic ensembles of classifiers applied to myocardium dysfunction recognition based on ECG signals. Swarm and evolutionary computation, 39, 192–208, (2018)
12. Przybyła-Kasperek M, Marfo KF. Neural Network Used for the Fusion of Predictions Obtained by the K-Nearest Neighbors Algorithm Based on Independent Data Sources. Entropy 23(12):1568, https://doi.org/10.3390/e23121568 (2021)
13. Russell, I.; Markov, Z. An introduction to the Weka data mining system. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, Seattle, WA, USA, 8–11 March, 742–742, (2017)
14. Yang, Q., Liu, Y., Cheng, Y., Kang, Y., Chen, T., Yu, H.: Federated learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 13(3), 1–207, (2019)
15. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST), 10(2), 1–19, (2019)
16. Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., Khazaeni, Y.: Bayesian nonparametric federated learning of neural networks. In International Conference on Machine Learning, 7252–7261, PMLR, (2019)
17. Zimmermann, A., Schmidt, R., Sandkuhl, K.: Multiple perspectives of digital enterprise architecture. In ENASE, 547–554, (2019)

Publication [P3]

Marfo K.F., Przybyła-Kasperek M. Radial basis function network for aggregating predictions of k-nearest neighbors local models generated based on independent data sets *Procedia Computer Science*, 207:3234–3243, 2022.
URL: http://dx.doi.org/10.1016/j.procs.2022.09.381.
DOI: 10.1016/j.procs.2022.09.381.
$\text{MEiN}_{2022} = 70$

Number of citations:

- according to Web of Science: 0
- according to Google Scholar: 8

Contributor	Description of main tasks
Kwabena Frimpong Marfo	- investigation - conceptualization and methodology - result analysis - implementation of proposed algorithm - manuscript: review and editing - visualization: creating all figures in manuscript
Małgorzata Przybyła-Kasperek	- conceptualization and methodology - result analysis - manuscript: original draft preparation - manuscript: review and editing - visualization: creating all figures in manuscript

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 207 (2022) 3228–3237

Procedia Computer Science

www.elsevier.com/locate/procedia

26th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2022)

Radial basis function network for aggregating predictions of *k*-nearest neighbors local models generated based on independent data sets

Kwabena Frimpong Marfo, Małgorzata Przybyła-Kasperek*

University of Silesia in Katowice, Institute of Computer Science,
Będzińska 39, 41-200 Sosnowiec, Poland

Abstract

In this article, a new classification method using neural networks for dispersed data from independent sources - local decision tables - is proposed. The presented method uses a modified *k*-nearest neighbors algorithm and a radial basis function neural network. Prediction vectors are generated by the modified *k*-nearest neighbors algorithm for all local decision tables, which are then passed to the neural network for a final decision. Comparative analysis of the error level and structural complexity were carried on the proposed method and a classification method which uses a modified *k*-nearest neighbors algorithm and a multi-layer perceptron. Results obtained shows that the proposed method generates unambiguous decisions and achieves lower error with less structural complexity as compared to the classification method which uses a modified *k*-nearest neighbors algorithm and the multi-layer perceptron.

Keywords: Federated learning; Radial basis functions; Neural networks; Independent data sources; *k*-nearest neighbors; Dispersed data.

1. Introduction

Traditional machine learning approaches require centralizing the training data that is possibly obtained from multiple data sources to a common server. However, in various application areas, data is usually isolated and independent, thus, combining them may be expensive due to communication costs, time sensitivity, or privacy concerns [13]. Consider as an instance, the health records of patients across various hospitals - such data is inherently sensitive and private, and with the emergence of laws and regulations of data privacy protection such as the GDPR [1], merging these health records from various hospitals to build an efficient machine learning model will be considered illegal if

* Corresponding author. Tel.: +48-32-368-97-56 ; fax: +48-32-368-97-60.
E-mail address: malgorzata.przybyla-kasperek@us.edu.pl

10.1016/j.procs.2022.09.381

performed without explicit consent from data owners. The issue of building a robust model by efficiently making use of data from multiple sources without violating data protection and privacy regulations brings in the idea of federated learning.

Federated Learning (FL) is a collaboratively decentralized privacy-preserving technology to overcome challenges of data silos and data sensibility [8]. As federated learning does not require moving data from independent sources to one central server, it makes this technique suitable for handling problems that involve dispersed data.

Another significant problem that appears in classification tasks based on dispersed data is the inconsistency and the heterogeneity of local data. If the data is collected in a completely independent manner, we cannot assume that all local data have the same sets of conditional attributes or the same sets of objects. The horizontal or vertical partition of data is rare in real applications. A more appropriate approach assumes that there may be common elements, but the sets are not equal. This statement applies to both the set of objects and the set of conditional attributes. This assumption can also be found in hybrid federated learning [14].

In the paper [11], a method for classification with the use of dispersed data was proposed. This method uses the k-nearest neighbors and a Multi-Layer Perceptron (MLP) classifiers. The k-nearest neighbors classifier is used for local data – prediction vectors are generated for all local tables, which are then transferred to a central server. Then a MLP makes the final decision based on all prediction vectors. In this way, we maintain data privacy as only prediction vectors are shared. The paper [11] presents that this approach gives unambiguous results, which cannot be said about other fusion methods such as the Majority Voting, the Borda Count method, the Sum Rule, the method based on decision templates and the method based on theory of evidence. Moreover, it was shown that the approach with neural networks achieves better quality of classification than the above-mentioned fusion methods. However, models built from this method does not scale well with noisy data – perturbations in the noise intensity and number of neurons in hidden layer significantly affects the performance of the method presented in [11].

This article focuses on the issue of building a classification method based on data from independent sources on the same issue, with the aim of proposing a new method that is more immune to perturbations of the parameters mentioned above. Specifically, the use of radial basis function neural networks (RBF neural network) in making the final decision based on all prediction vectors generated by the k-nearest neighbors classifier. RBF neural networks have proven to be powerful tools in universal function approximation [7] with a desirable feature of having a fast, linear learning algorithm capable of adequately handling complex non-linear mappings [5]. Moreover, RBF neural networks have been widely used to address problems with noisy data sets [4, 3, 6]. This motivates the use of RBF neural networks in place of MLP in the method presented in [11].

An important issue in the neural networks is its complexity. In real applications, we expect that the results generated by the model will be obtained in the shortest possible time. Thus, the structure and the complexity of the proposed model are also very important. Thus, in our research we also adopted some limitations to the complexity of the networks used. The two main conclusions of this paper are that the use of the RBF neural network approach provides, in most cases, better results than the MLP approach for the accepted upper bound of the complexity of the network's structure. In addition, RBF network allows to achieve good quality of classification with less complex structure.

The article is organized in the following way - Section 2 describes the classification approach of the dispersed data and the method of data generation. In Section 3, the results of the experiments are presented and analyzed. The article ends with conclusions.

2. Methodology

This section presents the method of classification for data from independent sources. The classification method proposed in this paper consists of two steps. The first step computes prediction vectors independently for all local decision tables using a modified k-nearest neighbors algorithm. We assume that a set of decision tables $D_{ag} = (U_{ag}, A_{ag}, d)$, $ag \in Ag$ from one discipline is available, where U_{ag} is the universe, a set of objects; A_{ag} is a set of conditional attributes; d is a decision attribute. In general, for each local decision table, a classifier is built. Ag is a set of classifiers, and ag is a single classifier. In the considered case, each of the classifiers $ag \in Ag$ will be the modified k-nearest neighbors classifier, so we will not get a model as such. For each local table and for each test object x, a probability vector over decision classes (denoted by $\mu_{ag}(x)$) is designated. The dimension of each vector $\mu_{ag}(x) = [\mu_{ag,1}(x), \ldots, \mu_{ag,c}(x)]$ is equal to the number of decision classes $c = card\{V^d\}$, where V^d is a set of decision

attribute values from all decision tables and $card\{V^d\}$ is the cardinality of this set. Each coefficient $\mu_{ag,j}(x)$ is determined using the k-nearest neighbors of the test object x belonging to a given decision class j and decision table D_{ag}. The Gower similarity measure is used in this approach. That is why the proposed model can be applied to data with mixed types of attributes: qualitative and quantitative. More precisely, the value $\mu_{ag,j}(x)$ is equal to the average of the Gower similarity between the k-nearest neighbors to the test object x from the j-th decision class and decision table D_{ag}.

In the second step, a neural network is used to develop a combined classification. In this stage, all prediction vectors from all local decision tables are transferred to a central server for the final step of computation. Specifically, for each test object, a global decision is generated with the use of a RBF neural network. The RBF neural network consists of three layers. This is because we do not want to increase the network's complexity. In addition, it has been verified that adding an additional layer does not bring significant improvement. An input layer consists of n neurons defined as the number of prediction vectors for one test object over all local decision tables and the number of decision classes, i.e $n = card\{Ag\} \times card\{V^d\}$. The number of neurons in the RBF layer (hidden layer) is the system's parameter that will be tested. Different values will be checked which are defined as a fraction of the number of neurons in the input layer. The output layer consists of neurons the number of which is equal to the number of decision classes. The radial basis function used in the RBF layer for each i-th neuron is the normalized Gaussian function given as

$$\Phi_i(y) = \exp\left[- \frac{\| y - c_i \|^2}{2\sigma_i^2} \right], \tag{1}$$

where y is the input vector, c_i is the center of the i-th neuron, σ_i is the i-th neurons bandwidth and $\| \cdot \|$ is the Euclidean norm. In our case, the vector y is the concatenation of prediction's vectors generated based on local decision tables for a given test object x. So y is the concatenation of vectors $\mu_{ag}(x), ag \in Ag$. The centers c_i which act as edges between the input layer and the RBF layer are approximated using the Lloyd's k-means algorithm [10]. The Gaussian width σ_i of the i-th neuron in the RBF layer is estimated as $\sigma_i = \frac{\rho_{\max}}{\sqrt{2n_H}}$ where $\rho_{\max}$ is the maximum distance between the chosen centers and n_H is the number of centers chosen which is equal to the number of neurons in the hidden layer. The back-propagation algorithm is used in the training of the RBF neuron network to determine the optimal weights and biases. The number of neurons in the RBF layer is adjusted as per the number of neurons in the input layer. In this study, the following values were analyzed $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 2\} \times$ the number of neurons in the input layer. For the output layer, the SoftMax activation function is used, as it has been shown to work well for multi-class problems [9]. As in the paper [11], the Adam optimizer as adaptive step size methods and the categorical cross-entropy loss function are used in this study. The structure of the RBF network is shown in Figure 1.

Since the prediction vectors were generated for the test objects using both the training and test objects, we employ a 10-fold cross stratified validation method for training and evaluating the model. At each iteration, the RBF network is trained using nine folds of the prediction vectors (equivalent to nine folds of the test data set), and the quality of classification of the model is evaluated on the last independent fold. This procedure is repeated five times and the results is averaged to determine the classification error level of the model. The pseudo-code that defines RBF neuron network is given in Listing 1.

3. Experimental results

The experiments were conducted on four data sets. Two of these data sets, Vehicle Silhouettes and the Lymphography (Large) were taken from the UC Irvine Machine Learning Repository [2]. Both data sets were randomly divided into two disjoint subsets, the training set (70% of objects) and the test set (30% of objects). The remaining two data sets are artificial data sets generated using the Weka [12] software. For this purpose, the RandomRBF method was used. This function, at first, randomly generates centers for each decision class. To make the set more difficult, several centroids were used to generate objects from one decision class: 100 centroids and 10 decision classes were used. Then to each center, a weight is randomly assigned and a central point per attribute, and a standard deviation. The new object is generated as follows. The center is selected according to the weights. Attribute values are randomly

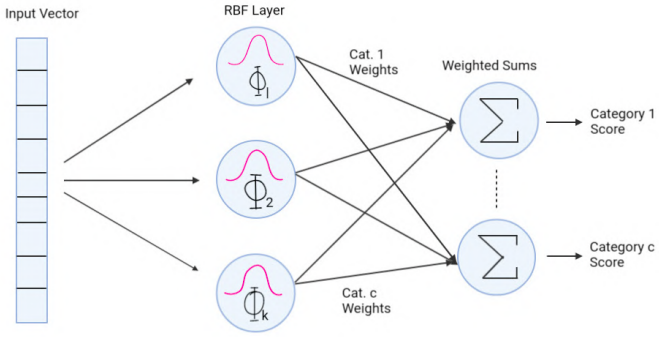


Fig. 1. Radial Basis Function Neural Network

Listing 1. Radial Basis Function Neural Network Model

```
def rbf_model(input_matrix, neurons:int, sigma:float):
  rbflayer = RBFLayer(neurons, initializer=InitCentersKMeans(input_matrix),
                betas=sigma, input_shape=(input_matrix.shape[1],))
  model = Sequential()
  model.add(rbflayer)
  model.add(Dense(classes, activation='softmax'))
  model.compile(loss='categorical_crossentropy', optimizer='adam',
          metrics=['error'])

  return model
```

generated and offset from the center. Then the vector is scaled so that its length equals a value sampled randomly from the Gaussian distribution of the center. The artificial data sets were also randomly, but in stratified way divided into training set (70% of the data) and testing set (30% of the data). To test the noise immunity of the model, a Gaussian noise level of mean = 0 and std = 0.2 was added to 30% of each attribute in the training set of both artificial data sets. The characteristics of the data sets are given in Table 1.

Table 1. Characteristics of Data sets

Data Set	# Training Set	# Test Set	# Conditional Attributes	# Decision Classes
Vehicle Silhouettes	592	254	18	4
Lymphography	104	44	18	4
Artificial Data 1	500	150	30	10
Artificial Data 2	500	150	50	10

Each of the data sets discussed above is stored in a single training table. Since the proposed model is dedicated to dispersed data, in the next step of data preparation, the data was dispersed. A different number of local decision tables (degree of dispersion) into which the training set was divided was considered, namely 3, 5, 7, 9, and 11 local tables. Attributes from original data set were randomly assigned to local tables in such a way that sets of attributes in local tables are not equal but they contain some common elements. All objects from the original table are included in each local table (with reduced attributes' set). However, objects' identifiers are not stored in local tables, so the objects between tables must be treated as different ones. The original form of the data cannot be restored based on local tables.

The error measure was used to evaluate the quality of classification. It is a fraction of the number of incorrectly classified objects by the total number of objects in the test set. The decisions generated by the proposed model are unambiguous; always only one decision class is generated.

As it was mentioned, the 10-fold cross-validation was used on the prediction vectors (which is equivalent to the test set) generated by the modified k-nearest neighbors classifier in order to train and evaluate the model. Each test was performed five times and the average of the obtained results are reported in the tables with the results below.

The experiments were carried out according to the following scheme:

- Generating vectors of predictions based on local tables using the k-nearest neighbors classifier. For each data set, three different values of the k parameter were tested, namely $k \in \{1, 5, 10\}$. These values were chosen arbitrarily, however $k = 10$ was chosen to observe if the proposed method is susceptible to generating ambiguous results. Comparative analysis for the optimal value of k was carried out for different degree of dispersion and data sets. For real data sets, Vehicle Silhouettes and Lymphography, different values of k parameters were optimal for different degrees of dispersion. Two groups of this degree can be distinguished; finer (7, 9, 11 local tables) and thicker (3, 5 local tables) dispersion. However, for Artificial Data sets $k = 1$ was always the optimal value.
- Generating a global decision using RBF neural network with one hidden layer and different number of neurons in the hidden layer. For each data set, the following number of neurons in the hidden layer were tested: $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 2\} \times$ the number of neurons in the input layer. Such values were adopted to reduce the complexity of the system and keep the training time of the model short. It was also caused by the small size of the data sets (especially the size of the test set is important here).
- Generating a global decision using the model proposed in [11] with the MLP neural network (MLP network).

Comparison of experimental results was made in terms of:

- The quality of classification for different values of the parameter k;
- The quality of classification for different numbers of neurons in the hidden layer;
- The quality of classification of the proposed dispersed classification model with the RBF network versus the dispersed classification model with the MLP network.

3.1. Comparison of experimental results for different values of k parameter and different numbers of neurons in the hidden layer

In Tables 2, 3, 4 and 5 the results obtained for the new proposed method using the RBF network and different data sets are presented. The rows show the results for different values of k parameter of the nearest neighbors algorithm and different dispersion versions. The columns present the results obtained with using different number of neurons in the hidden layer of the RBF network. The best result in each row is given in bold.

The main conclusions drawn from the results are as follows. Considering the Vehicle data set (results from Table 2), we observe that, among the variations of nearest neighbors i.e $k \in \{1, 5, 10\}$, $k = 1$ produces better results for thicker dispersion whiles for finer dispersion, $k = 5$ achieves better results. For the Lymphography data set, we observe from Table 3 that, for finer dispersion, $k = 5$ achieves better results, while $k = 10$ achieves better results for finer dispersion. We emphasize for Artificial Data sets 1 and 2 (Tables 8 and 9), $k = 1$ was always the optimal error value. For all considered data sets, the minimal error, in most cases, is achieved with less structural complexity. In other words, the minimal error is not obtained for the most complex structure of neural network that was analyzed (the largest number of neurons in the hidden layer).

3.2. Comparison of results for the proposed model with RBF network versus the model with MLP network

In Tables 6, 7, 8 and 9 the corresponding results obtained for the MLP network approach (from the paper [11]) are presented. The best result in each row is given in bold. Also, the best, the lowest value of error measure obtained for each k parameter value and each dispersion version for the network RBF and network MLP approaches were compared in Table 10. Again, the better result of the two approaches is shown in bold here.

Table 2. Results of classification error e for the Vehicle data set and RBF network. Designation I is used for the number of neurons in the input layer.

k-NN	No. of local tables	No. of neurons in hidden layer										
		0.1×I	**0.2×I**	**0.3×I**	**0.4×I**	**0.5×I**	**0.6×I**	**0.7×I**	**0.8×I**	**0.9×I**	**1×I**	**2×I**
		e	e	e	e	e	e	e	e	e	e	e
1-NN	3	0.784	0.618	0.491	0.427	0.404	0.338	0.317	0.287	0.280	0.281	**0.265**
	5	0.632	0.453	0.412	0.360	0.294	0.303	0.291	**0.263**	0.27	**0.263**	0.265
	7	0.583	0.484	0.41	0.365	0.317	0.311	0.307	0.292	0.292	**0.286**	0.295
	9	0.456	0.389	0.317	0.293	0.28	0.293	0.283	**0.277**	0.280	**0.277**	0.306
	11	0.485	0.410	0.363	0.327	0.317	0.330	0.322	0.296	**0.289**	0.299	0.300
5-NN	3	0.787	0.628	0.486	0.463	0.407	0.360	0.328	0.294	0.293	0.282	**0.260**
	5	0.641	0.520	0.465	0.349	0.333	0.304	0.291	0.284	0.269	0.267	**0.242**
	7	0.597	0.473	0.415	0.364	0.345	0.303	0.304	**0.293**	0.303	0.294	0.307
	9	0.494	0.436	0.332	0.309	**0.282**	0.296	0.294	0.286	0.295	0.293	0.312
	11	0.479	0.407	0.36	0.336	0.333	0.321	0.299	0.302	0.299	**0.291**	0.300
10-NN	3	0.783	0.631	0.477	0.451	0.401	0.356	0.343	0.314	0.306	0.306	**0.281**
	5	0.642	0.502	0.463	0.378	0.362	0.333	0.330	0.304	0.309	0.284	**0.257**
	7	0.609	0.476	0.395	0.370	0.357	0.335	0.303	0.308	**0.300**	0.301	0.307
	9	0.512	0.414	0.362	0.341	0.297	0.294	0.294	0.303	**0.289**	0.297	0.312
	11	0.519	0.407	0.38	0.347	0.337	0.338	0.319	0.320	**0.304**	0.323	0.319

Table 3. Results of classification error e for the Lymphography data set and RBF network. Designation I is used for the number of neurons in the input layer.

k-NN	No. of local tables	No. of neurons in hidden layer										
		0.1×I	**0.2×I**	**0.3×I**	**0.4×I**	**0.5×I**	**0.6×I**	**0.7×I**	**0.8×I**	**0.9×I**	**1×I**	**2×I**
		e	e	e	e	e	e	e	e	e	e	e
1-NN	3	0.622	0.576	0.542	0.52	0.501	0.467	0.371	0.343	0.333	0.322	**0.298**
	5	0.544	0.509	0.433	0.351	0.281	0.286	0.289	0.266	0.268	**0.239**	0.257
	7	0.616	0.479	0.437	0.400	0.402	**0.367**	0.381	0.382	0.399	0.378	0.393
	9	0.533	0.481	0.379	0.38	0.388	**0.341**	0.366	0.397	0.386	0.359	0.380
	11	0.487	0.296	0.286	0.275	0.270	0.308	0.304	**0.264**	0.329	0.336	0.326
5-NN	3	0.617	0.614	0.531	0.341	0.375	0.302	0.308	0.277	0.292	**0.242**	0.258
	5	0.551	0.477	0.323	0.272	0.252	0.210	0.204	**0.184**	0.209	0.209	0.249
	7	0.589	0.427	0.375	0.343	0.347	0.352	0.364	**0.338**	0.34	0.375	0.346
	9	0.507	0.427	0.369	0.324	0.295	0.287	0.287	**0.267**	0.308	0.334	0.316
	11	0.482	0.298	0.266	0.269	**0.228**	0.282	0.274	0.291	0.27	0.29	0.299
10-NN	3	0.634	0.595	0.475	0.419	0.394	0.385	0.356	0.278	0.291	0.286	**0.240**
	5	0.553	0.457	0.348	0.289	0.26	0.238	0.223	0.221	**0.212**	0.225	**0.212**
	7	0.542	0.425	0.355	0.334	0.318	**0.313**	0.358	0.334	**0.313**	0.338	0.332
	9	0.513	0.414	0.337	0.334	0.322	0.312	0.335	0.324	0.313	**0.31**	0.338
	11	0.494	0.275	0.228	0.234	**0.218**	0.240	0.268	0.252	0.252	0.252	0.269

The main conclusions from the comparison of the results are as follows. For the Vehicle data set, observing from Tables 2 and 6, we see that:

- For all levels of k and all levels of data dispersion, the RBF network achieves better results than the MLP network.

Table 4. Results of classification error *e* for the Artificial Data 1 and RBF network. Designation I is used for the number of neurons in the input layer.

k-NN	No. of local tables	No. of neurons in hidden layer 0.1×I *e*	0.2×I *e*	0.3×I *e*	0.4×I *e*	0.5×I *e*	0.6×I *e*	0.7×I *e*	0.8×I *e*	0.9×I *e*	1×I *e*	2×I *e*
1-NN	3	0.687	0.422	0.267	0.131	0.093	**0.040**	0.047	0.053	0.049	0.044	0.051
	5	0.558	0.247	0.116	0.042	0.051	0.049	**0.038**	0.049	0.044	0.062	0.08
	7	0.422	0.151	0.084	0.082	**0.08**	0.082	0.087	0.082	0.089	0.098	0.138
	9	0.398	0.164	0.100	0.093	**0.082**	0.091	0.093	0.113	0.116	0.153	0.231
	11	0.480	0.173	0.176	0.140	0.160	0.158	**0.138**	0.173	0.171	0.187	0.331

Table 5. Results of classification error *e* for the Artificial Data 2 and RBF network. Designation I is used for the number of neurons in the input layer.

k-NN	No. of local tables	No. of neurons in hidden layer 0.1×I *e*	0.2×I *e*	0.3×I *e*	0.4×I *e*	0.5×I *e*	0.6×I *e*	0.7×I *e*	0.8×I *e*	0.9×I *e*	1×I *e*	2×I *e*
1-NN	3	0.698	0.469	0.256	0.096	0.071	0.051	0.062	0.060	0.033	**0.031**	0.042
	5	0.467	0.193	0.113	**0.018**	0.038	0.024	0.024	0.040	0.031	0.036	0.050
	7	0.393	0.149	0.060	0.038	0.042	0.042	0.044	0.051	0.040	**0.036**	0.102
	9	0.26	0.096	**0.016**	0.018	0.024	0.031	0.036	0.036	0.047	0.076	0.098
	11	0.127	**0.024**	**0.024**	0.027	0.029	0.051	0.049	0.067	0.049	0.071	0.196

Table 6. Results of classification error *e* for the Vehicle data set and MLP network. Designation I is used for the number of neurons in the input layer.

k-NN	No. of local tables	No. of neurons in hidden layer 0.1×I *e*	0.2×I *e*	0.3×I *e*	0.4×I *e*	0.5×I *e*	0.6×I *e*	0.7×I *e*	0.8×I *e*	0.9×I *e*	1×I *e*	2×I *e*
1-NN	3	0.692	0.612	0.553	0.469	0.455	0.464	0.423	0.414	0.397	0.416	**0.329**
	5	0.575	0.573	0.490	0.470	0.416	0.38	0.395	0.361	0.349	0.352	**0.317**
	7	0.637	0.508	0.456	0.437	0.414	0.414	0.393	0.371	0.382	0.375	**0.356**
	9	0.561	0.516	0.421	0.417	0.414	0.372	0.373	0.356	0.34	0.351	**0.324**
	11	0.589	0.515	0.437	0.437	0.419	0.409	0.382	0.383	0.363	0.364	**0.352**
5-NN	3	0.648	0.6	0.549	0.526	0.481	0.44	0.445	0.427	0.405	0.391	**0.344**
	5	0.632	0.532	0.496	0.414	0.422	0.402	0.402	0.373	0.376	0.366	**0.329**
	7	0.607	0.489	0.455	0.423	0.430	0.421	0.376	0.377	0.367	0.347	**0.332**
	9	0.536	0.500	0.416	0.368	0.359	0.339	0.355	0.325	0.318	0.319	**0.301**
	11	0.604	0.455	0.409	0.384	0.373	0.358	0.370	0.346	0.353	0.327	**0.326**
10-NN	3	0.701	0.587	0.535	0.524	0.488	0.491	0.443	0.404	0.410	0.405	**0.352**
	5	0.605	0.529	0.47	0.468	0.426	0.435	0.392	0.371	0.383	0.361	**0.334**
	7	0.553	0.487	0.492	0.407	0.402	0.386	0.36	0.371	0.377	0.363	**0.326**
	9	0.566	0.473	0.423	0.391	0.376	0.359	0.329	0.327	0.326	0.342	**0.296**
	11	0.607	0.458	0.392	0.385	0.374	0.36	0.347	0.367	0.341	**0.319**	0.330

- For almost all data dispersion levels, the RBF network achieves minimal error with less complex structure (fewer number of neurons in RBF Layer) as compared to the MLP network.

For the Lymphography data set, we observe the following from Tables 3 and 7

Table 7. Results of classification error e for the Lymphography data set and MLP network. Designation I is used for the number of neurons in the input layer.

k-NN	No. of local tables	No. of neurons in hidden layer										
		0.1×I	**0.2×I**	**0.3×I**	**0.4×I**	**0.5×I**	**0.6×I**	**0.7×I**	**0.8×I**	**0.9×I**	**1×I**	**2×I**
		e	e	e	e	e	e	e	e	e	e	e
1-NN	3	0.604	0.536	0.502	0.447	0.427	0.483	0.420	0.392	0.326	0.355	**0.309**
	5	0.499	0.412	0.322	0.289	0.276	0.274	0.242	0.244	0.25	**0.230**	0.252
	7	0.485	0.419	0.413	0.351	0.374	0.364	0.394	0.369	0.361	**0.322**	0.333
	9	0.409	0.302	0.305	0.291	0.300	**0.288**	0.3	0.301	0.297	0.311	0.309
	11	0.384	0.313	0.322	0.326	0.332	0.319	0.345	0.338	**0.311**	0.344	0.312
5-NN	3	0.562	0.547	0.456	0.427	0.418	0.372	0.345	0.321	0.293	0.284	**0.277**
	5	0.424	0.378	0.318	0.245	0.238	**0.196**	0.224	0.221	0.204	0.217	0.2592
	7	0.42	0.389	0.333	0.393	0.329	0.334	**0.315**	0.333	0.341	0.342	0.327
	9	0.325	0.307	**0.270**	0.289	0.271	0.297	0.297	0.301	0.278	0.288	0.305
	11	0.337	0.261	0.258	**0.252**	0.294	0.287	0.269	0.310	0.279	0.268	0.294
10-NN	3	0.599	0.543	0.477	0.414	0.389	0.358	0.347	0.351	0.299	0.321	**0.277**
	5	0.477	0.402	0.261	0.231	0.253	0.223	**0.203**	0.204	0.218	0.214	0.232
	7	0.439	0.366	0.351	0.297	0.316	0.302	0.298	0.287	**0.275**	0.297	0.295
	9	0.383	0.279	0.259	0.267	0.26	**0.248**	0.259	0.26	0.254	0.267	0.269
	11	0.349	0.245	**0.235**	0.249	0.245	0.2641	0.254	0.257	0.245	0.269	0.263

Table 8. Results of classification error e for the Artificial Data 1 and MLP network. Designation I is used for the number of neurons in the input layer.

k-NN	No. of local tables	No. of neurons in hidden layer										
		0.1×I	**0.2×I**	**0.3×I**	**0.4×I**	**0.5×I**	**0.6×I**	**0.7×I**	**0.8×I**	**0.9×I**	**1×I**	**2×I**
		e	e	e	e	e	e	e	e	e	e	e
1-NN	3	0.769	0.742	0.624	0.604	0.540	0.531	0.456	0.400	0.429	0.347	**0.153**
	5	0.764	0.627	0.613	0.518	0.418	0.389	0.273	0.264	0.224	0.184	**0.087**
	7	0.727	0.609	0.549	0.449	0.371	0.291	0.258	0.216	0.220	0.196	**0.111**
	9	0.782	0.689	0.556	0.451	0.398	0.316	0.262	0.251	0.242	0.222	**0.118**
	11	0.796	0.66	0.613	0.489	0.384	0.378	0.302	0.316	0.287	0.249	**0.202**

Table 9. Results of classification error e for the Artificial Data 2 and MLP network. Designation I is used for the number of neurons in the input layer.

k-NN	No. of local tables	No. of neurons in hidden layer										
		0.1×I	**0.2×I**	**0.3×I**	**0.4×I**	**0.5×I**	**0.6×I**	**0.7×I**	**0.8×I**	**0.9×I**	**1×I**	**2×I**
		e	e	e	e	e	e	e	e	e	e	e
1-Nearest Neighbor	3	0.762	0.658	0.511	0.462	0.387	0.351	0.291	0.202	0.209	0.120	**0.040**
	5	0.691	0.571	0.436	0.307	0.220	0.167	0.102	0.082	0.049	0.044	**0.024**
	7	0.607	0.451	0.329	0.193	0.116	0.073	0.049	0.04	0.033	**0.022**	0.031
	9	0.598	0.409	0.196	0.124	0.051	0.056	0.031	**0.024**	0.033	0.027	0.031
	11	0.582	0.329	0.124	0.051	0.042	0.04	0.031	**0.022**	0.027	**0.022**	0.027

- For $k = 1$, for all dispersion levels, though the RBF network performs same or worse than the MLP network, it achieves its minimal error with less complex structure.
- For $k = 5$, for almost all dispersion levels (except for 7 local tables) the RBF network performs better and achieves its minimal error with less complex structure as compared to the MLP network.

- For $k = 10$, the RBF network performs comparatively worse both in terms of error level and structure complexity.
- Nevertheless, if we choose the best results among all the values of k parameter, it turns out that the best results were obtained by the RBF network in all cases of dispersion except for 7 local tables.

For Artificial Data 1, we observe from Tables 4 and 8 that for all levels of k and all levels of data dispersion, the RBF network achieves better results than the MLP network. For Artificial Data 2, we observe from Tables 5 and 9, for all dispersion levels except for 7 local tables, the RBF network achieves better results and also achieves its minimal error with less complex structure as compared to the MLP for all levels of dispersion. Thus, in general statement, for all analyzed data sets, the RBF network approach gives better results than the MLP network approach both in terms of error measure and network structure complexity.

Table 10. Comparison of minimal classification errors e obtained for the approaches with RBF network and with MLP network. Designation I is used for the number of neurons in the input layer.

	No. of local tables	**hidden layer RBF**	**Minimal Error e RBF** e	**MLP** e	**hidden layer MLP**	**hidden layer RBF**	**Minimal Error e RBF** e	**MLP** e	**hidden layer MLP**
			Vehicle				**Lymphography**		
1-NN	3	2×I	**0.265**	0.329	2×I	2×I	**0.298**	0.309	2×I
	5	{0.8, 1}×I	**0.263**	0.317	2×I	1×I	0.239	**0.230**	1×I
	7	1×I	**0.286**	0.356	2×I	0.6×I	0.367	**0.322**	1×I
	9	{0.8, 1}×I	**0.277**	0.324	2×I	0.6×I	0.341	**0.288**	0.6×I
	11	0.9×I	**0.289**	0.352	2×I	0.8×I	**0.264**	0.311	0.9×I
5-NN	3	2×I	**0.260**	0.344	2×I	1×I	**0.242**	0.277	2×I
	5	2×I	**0.242**	0.329	2×I	0.8×I	**0.184**	0.196	0.6×I
	7	0.8×I	**0.293**	0.332	2×I	0.8×I	0.338	**0.315**	0.7×I
	9	0.5×I	**0.282**	0.301	2×I	0.8×I	**0.267**	0.270	0.3×I
	11	1×I	**0.291**	0.326	2×I	0.5×I	**0.228**	0.252	0.4×I
10-NN	3	2×I	**0.281**	0.352	2×I	2×I	**0.240**	0.277	2×I
	5	2×I	**0.257**	0.334	2×I	2×I	0.212	**0.203**	0.7×I
	7	0.9×I	**0.300**	0.336	2×I	0.9×I	0.313	**0.275**	0.9×I
	9	0.9×I	**0.289**	0.296	2×I	1×I	0.310	**0.248**	0.6×I
	11	0.9×I	**0.304**	0.319	1×I	0.5×I	**0.218**	0.235	0.3×I
			Artificial Data 1				**Artificial Data** 2		
1-NN	3	0.6×I	**0.040**	0.153	2×I	1×I	**0.031**	0.040	{0.8, 2}×I
	5	0.7×I	**0.038**	0.087	2×I	0.4×I	**0.018**	0.024	2×I
	7	0.5×I	**0.080**	0.111	2×I	1×I	0.036	**0.022**	1×I
	9	0.5×I	**0.082**	0.118	2×I	0.3×I	**0.016**	0.024	0.8×I
	11	0.7×I	**0.138**	0.202	2×I	{0.2, 0.3}×I	0.024	**0.022**	{0.8, 1}×I

In order to investigate the significance in differences of error measure obtained for the proposed dispersed classification model with the RBF network and the dispersed classification model with the MLP network all results from Table 10 were used. Two dependent samples were created – one containing the results for RBF network and one containing the results for MLP network. Each sample had the cardinality equal to 40 observations – results obtained for different data sets, number of local tables and k parameter in the nearest neighbors algorithm. The Wilcoxon test confirmed that differences in error rate in these two groups are significant, with a level of $p = 0.0003$. Additionally, comparative box-plot chart for the values of error measure was created (Figure 2). The graph also confirms that the approach RBF network generates better results.

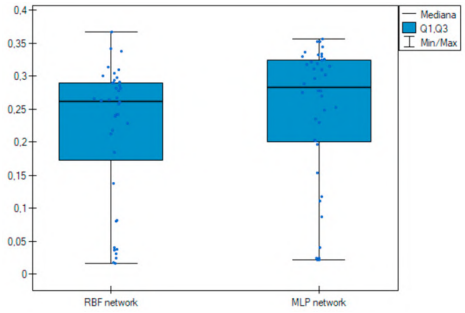


Fig. 2. Box-plot chart with (Median, the first quartile—Q1, the third quartile—Q3) the value of error measure e for the approaches with RBF network and with MLP network.

4. Conclusions

In this paper, a classification method based on a modified k-nearest neighbors and a RBF network classifiers for generating unambiguous decisions based on dispersed data from independent sources is proposed. The paper makes comparison between results from the proposed method and results from classification method based on a modified k-nearest neighbors and a MLP network classifiers. With the use of data from the UC Irvine Machine Learning Repository and artificially generated data sets, comparative analysis was carried out on various dispersion levels of data as well as the structural complexity of both RBF and MLP networks. The comparative analysis showed that the RBF proposed method achieves better results with a less structural complexity as compared to the method proposed in [11]. The presented studies focuses on showing that the proposed approach has potential and will be developed in the future. The main limitation of the proposed method is the propensity of the Lloyd's algorithm getting stuck in local minima, which affects the overall predictive quality of models generated using the proposed method. In future research, it is planned to adopt some meta-heuristic techniques to address this caveat. Also, It would be very interesting to average the prediction vectors from all local tables before passing to the neural network. This way, we reduce the size of input vectors, which might significantly improve the completion time of the proposed method.

References

[1] Albrecht, J. P. (2016): How the GDPR will change the world. Eur. Data Prot. L. Rev., 2, 287, 2016.
[2] Asuncion, A., Newman, D.J.: UCI Machine Learning Repository; University of Massachusetts Amherst: Amherst, MA, USA, 2007. Available online: https://archive.ics.uci.edu (accessed on 25 April 2022).
[3] Bambang, R. T., Anggono, L., Uchida, K.: DSP based RBF neural modeling and control for active noise cancellation. In Proceedings of the IEEE internatinal symposium on intelligent control, IEEE, 460-466., 2002.
[4] Bernier, J. L., Daz, A. F., Fernandez, F. J., Caas, A., Gonzlez, J., Martn-Smith, P., Ortega, J.: Assessing the noise immunity and generalization of radial basis function networks. Neural Processing Letters, 18(1), 35-48, 2003.
[5] Dash, C. S. K., Behera, A. K., Dehuri, S., Cho, S. B.: Radial basis function neural networks: a topical state-of-the-art survey. Open Computer Science, 6(1), 33-63, 2016,.
[6] Debakla, M., Djemal K., Benyettou M.: A Novel Approach for Medical Images Noise Reduction Based RBF Neural Network Filter. J. Comput, 10(2), 68-80, 2015.
[7] Haykin, S., Network, N.: A comprehensive foundation. Neural networks, 2(2004), 41.
[8] Li, L., Fan, Y., Tse, M., Lin, K. Y.: A review of applications in federated learning. Computers & Industrial Engineering, 149, 106854,2020
[9] Li, X., Xin L., Deng P., and Dongxiao Z.: On the learning property of logistic and softmax losses for deep neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 04, pp. 4739-4746. 2020.
[10] Mashor, M. Y.: Improving the performance of k-means clustering algorithm to position the centers of RBF network. International Journal of the Computer, The Internet and Management, 6(2), 121–124, 1998.
[11] Przybyła-Kasperek, M., Marfo, K. F.: Neural Network Used for the Fusion of Predictions Obtained by the K-Nearest Neighbors Algorithm Based on Independent Data Sources. Entropy, 23(12), 1568, 2021.
[12] Russell, I., Markov, Z.: An introduction to the Weka data mining system. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, Seattle, WA, USA, (2017).
[13] Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., Khazaeni, Y.: Bayesian nonparametric federated learning of neural networks. In International Conference on Machine Learning, 7252-7261, PMLR, 2019.
[14] Zhu, H., Zhang, H., Jin, Y.: From federated learning to federated neural architecture search: a survey. Complex & Intelligent Systems, 7(2), 639–657, 2021.

Publication [P4]

Marfo K.F., Przybyła-Kasperek M. Radial basis function neural network with a centers training stage for prediction based on dispersed image data *International Conference on Computational Science*, 10476:89-103, 2023.
URL: https://link.springer.com/chapter/10.1007/978-3-031-36027-5_7.
DOI: 10.1007/978-3-031-36027-5_7.
$\text{MEiN}_{2023} = 140$

Number of citations:

- according to Web of Science: 0
- according to Google Scholar: 4

Contributor	Description of main tasks
Kwabena Frimpong Marfo	- investigation - conceptualization and methodology - result analysis - implementation of proposed algorithm - manuscript: review and editing - visualization: creating all figures in manuscript
Małgorzata Przybyła-Kasperek	- conceptualization and methodology - result analysis - manuscript: original draft preparation - manuscript: review and editing - visualization: creating all figures in manuscript

Radial Basis Function Neural Network with a Centers Training Stage for Prediction Based on Dispersed Image Data

Kwabena Frimpong Marfo[1][0000−0003−2226−9097] and Małgorzata Przybyła-Kasperek[1][0000−0003−0616−9694]

University of Silesia in Katowice, Institute of Computer Science, Będzińska 39, 41-200 Sosnowiec, Poland
{kwabena.marfo,malgorzata.przybyla-kasperek}@us.edu.pl

Abstract. Neural networks perform very well on difficult problems such as image or speech recognition as well as machine text translation. Classification based on fragmented and dispersed data representing certain properties of images or computer's vision is a complex problem. Here, the suitability of a Radial Basis Function (RBF) neural network was evaluated using fragmented data in the problem of recognizing objects in images. The great difficulty of the considered problem is, there is not images data as such but only data on some properties of images stored in a dispersed form. More specifically, it was demonstrated that applying a $k-$nearest neighbors classifier in the first step to generate predictions based on fragmented data, and then using a RBF neural network to learn how to correctly recognize the systems of generated predictions for making a final classification is a good approach for recognizing objects in images. An additional step of training the weights (centers) between the input and hidden layers of a RBF network was proposed. In general, this investigation demonstrates that adding this step significantly improves the correctness of recognizing objects in images.

Keywords: Radial Basis Function Neural Network · Dispersed Data · Image Data Processing.

1 Introduction

Object detection in images is very important in today's world and many applications such as surveillance systems can be found. Examples of important uses include: recognizing types of vehicles in road traffic [18], recognizing types of objects in satellite images [12], or even recognizing components on a production line [20, 2]. In literature, we can find numerous applications of neural networks for processing images and recognizing objects in images [21, 19, 1]. These are mainly applications of convolutional neural networks. A very interesting approach where neural networks were used for recognizing handwriting can be found in [14]. In [5], the generative adversarial network (GAN) was used to generate images. Paper [15] provided an overview of very interesting approaches that used neural networks to generate artistic patterns.

It is very rare to find studies that deal with recognizing images that are available in fragmentary form. This problem can be considered as a set of images obtained from the perspective of several cameras, each of which observes the object from different angles [16]. In such a case – when the image data is fragmented, recognizing the object in the photo is more difficult. In the paper [7], the approach of assembling fragments of photos and matching parts to merge into one can be found. This approach required that the fragmented data do not overlap, however, in this study, fragmentation concerns the dispersion of information about the recognized object with shared fragments and does not mean disjointed information. We also assume that we do not have images as such, nor its fragments, but only the characteristics extracted from these images. These may be the average values recorded in a certain area of pixels or certain shape characteristics such as the length and width depicted in the image. We also assume that these data are available in tabular form – a set of decision tables. However, the data may overlap, i.e. there may be common conditional attributes in several decision tables. In addition, in the training set which comprises a set of local tables, there may appear fragmentary characteristics for the same physical object or for other objects belonging to the same decision class. The focus of this study was not on image fusion but on object recognition, assigning the correct decision class for an object that is seen in a fragmented way.

Other approaches used to recognize fragmented images can also be found in literature. The paper [22] proposed the use of hidden Markov models for gesture recognition based on fragmentary vision. In the paper [3], fuzzy rules were used for fragmented handwritten digit recognition.

This paper proposes the use of a Radial Basis Function (RBF) neural network with a centers training stage in combination with the $k-$nearest neighbors algorithm to classify objects observed in images based on fragmented characteristics – data stored as a set of local decision tables. For this purpose, three problems were considered: classification of car type based on photo's characteristics, classification of land type based on satellite images, and classification of bean type based on fragmentary computer vision. To the best of our knowledge, such an issue has not been studied before in literature. In this study research results, comparisons and statistical tests are presented. It has been justified that the proposed approach gives much better results than the RBF networks without a centers training stage as well as the baseline approach that uses a heterogeneous ensemble of classifiers.

The paper is organized as follows. In Section 2, the proposed classification model using a RBF neural network is described. The algorithm's description and the discussion about the key features of the proposed approach are given. Section 3 addresses the datasets that were used and presents the conducted experiments and discussion on obtained results. Section 4 is on conclusions and future research plans.

2 Model and methods

When data is stored in dispersed form, aggregating local tables into a table becomes a difficult task due to data inconsistencies. To overcome this, we consider local data separately, look for patterns in them and find a way to combine the dependencies already discovered into a single piece. More formally, we assume that some characteristics of images are available in dispersed form – in the form of a set of local tables. We assume that a set of decision tables $D_i = (U_i, A_i, d)$, $i \in \{1, \ldots, n\}$ is available, where U_i is the universe comprising a set of objects – images; A_i is a set of conditional attributes – some features that describe the image; d is a decision attribute – object shown in the image. Objects and attributes in local tables can be different, however, some objects may be common among local tables as we do not impose any restrictions here.

This is a case where different sensors capture features of an image based on the object identified in the image. In a real-life situation, it could be pictures taken at different angles of the same object by different cameras. Thus, local tables will be generated (one local table for each camera) with different sets of conditional attributes (but some may be shared). Another instance could be cameras set up in different locations in a city where each camera takes pictures of vehicles on the road and based on the features identified, the type of vehicle is determined. In this situation, same object could be recognized by different cameras, but most of the objects identified will be different.

Since aggregation of these local tables into a single table is not immediately possible, one possible approach that can be used is to generate classifiers based on each local table separately. For this purpose, the $k-$nearest neighbors classifier is chosen, as it has low computational complexity and is a suitable method for image classification based on local features [8]. We expect objects of one type in images to have similar features. To calculate the distances between the test objects and the objects in the local tables, the Gower measure is used [13]. This measure allows to compute the distance even when there are attributes of different types (quantitative, qualitative, binary) and from different ranges in the decision table (it does not require normalization or standardization). In addition, it should be noted that for the test objects, values of all attributes occurring in the local tables should be known. Each base classifier makes its classification using a subset of all attributes – more strictly, a classifier i that is built based on a decision table D_i uses the set of attributes A_i. A modification of the $k-$nearest neighbors classifier is proposed, i.e. instead of generating a prediction from the abstract level (a decision class most frequent in the neighborhood), a prediction from the measurement level is generated. That is, a classifier i generates a probability vector over decision classes for a test object x (denoted by $\mu_i(x)$). The dimension of vector $\mu_i(x) = [\mu_{i,1}(x), \ldots, \mu_{i,c}(x)]$ is equal to the number of decision classes $c = card\{V^d\}$, where V^d is a set of decision attribute values (a set of the types of objects in the image), $card\{V^d\}$ is the cardinality of this set. Each coefficient $\mu_{i,j}(x)$ is determined using the k-nearest neighbors of the test object x belonging to a given decision class j and decision table D_i. In this way, the information we get from the base classifiers is more complete. These

are average similarities determined from fragmented data for each decision class (i.e. objects that may appear in the image).

To summarize the previous step, the base classifiers for the test object x generates n prediction vectors $\mu_i(x)$, $i \in \{1, \ldots, n\}$, each vector is c dimensional, i.e. a composite of similarities to decision classes obtained based on fragmented data. The task of recognizing the correct decision class based on such vectors – identifying the object in the image – is a difficult task. For this purpose the Radial Basis Function (RBF) neural network is used.

We make use of a RBF neural network because they have the property of separating spaces that are not linearly separable. This is achieved by transforming the input vectors into a new feature space, where classes are linearly separable. For this to be possible, there must be more neurons in the hidden layer than in the input layer. In the considered problem, the input vector is formed through the prediction vectors generated by the base classifiers. So, more formally, a vector will be created $[\mu_1(x), \ldots, \mu_n(x)]$ for the test object x. Its dimension is equal to $n \cdot c$ because we have n base classifiers, each generating a c dimensional vector. Thus, we have $n \cdot c$ neurons in the input layer. RBF neural networks always contain only one hidden layer, thus, there are no problems with determining the appropriate number of hidden layers for a given problem. Each neuron in the hidden layer has a parameter called center. Formally, the $k-$th neuron in the hidden layer has the center c_k. The center is interpreted as a representative of a certain group of objects. RBF networks are similar to the $k-$nearest neighbors approach. However, instead of eliminating distant objects from the classification process (as it is done in the $k-$nearest neighbors classifier), this time we simply reduce the influence of distant objects on the output of the neural network, but still use them in the classification process. We obtain this property by using the Gaussian function as the activation function for each of the neurons in the hidden layer. The more the input vector is similar to the center of the neuron in the hidden layer, the greater its influence on the output of the neural network. Let us denote the vector given as the input of the network in the input layer by y. Then for the $k-$th neuron of the hidden layer the Gaussian function is as follows

$$\Phi_k(y) = \exp\left[- \frac{\| y - c_k \|}{2\sigma_k^2} \right], \tag{1}$$

where c_k is the center of the $i-$th neuron, σ_k is the $k-$th neuron's bandwidth and $\| \cdot \|$ is the Euclidean norm. The Gaussian width σ_k of the $k-$th neuron in the hidden layer was estimated as $\sigma_k = \frac{\rho \max}{2 \cdot n_H}$ where $\rho_{\max}$ is the maximum distance between the chosen centers and n_H is the number of neurons in the hidden layer. The weights and biases assigned to the connections of neurons from the hidden layer to the output layer in the RBF network are trained using the back-propagation algorithm.

A very important issue in RBF networks is the appropriate determination of the center of neurons in the hidden layer. This is usually done by a clustering algorithm realized on the training set (this is implemented before the training of the network). Usually the $k-$means clustering algorithm is used and the centroids

determined by this algorithm are used as centers connecting the input layer to the hidden layer neurons. The Lloyd's $k-$means algorithm – a modification of the $k-$means algorithm is also very often used. In the study, one of the approaches analyzed is the Lloyd's $k-$means algorithm for determining centers. But more interesting is the proposed approach in which instead of designating centers only once (determined by the Lloyd's $k-$means algorithm), a step of training these centers is used. This is implemented as follows.

- In the first stage, the Lloyd's $k-$means algorithm is used to determine the centers which serves as connections between the input layer and the hidden layer.
- After, random values are assigned as weights and biases, which serve as connections between the hidden layer and the output layer.
- Then, training the RBF network begins, however, here the propagation of the error is extended up to the centers. This way, the centers are trained iteratively together with the weights and biases using the back-propagation method.

The pseudo-code for the proposed RBF neural network is given below in Listing 2.

RBF Neural Network with a Centers Training Stage

```
# X: matrix of prediction vectors over all local tables
# centers: array of values determined by the Lloyd's algorithm
# neurons: number of neurons in the hidden layer
# sigma: Gaussian width

def RBFModel(X:array, centers:array, neurons:int, sigma:float):
  model = Sequential()
  model.add(Flatten(input_shape=(X.shape[1],)))
  model.add(RBFLayer(units= neurons, gamma=sigma))
  model.add(Dense(classes, activation='softmax'))
  model.layers[1].set_weights([centers])
  model.compile(loss='categorical_crossentropy',
  optimizer='adam',metrics=['accuracy'])
  return model
```

The above describes how to build and train the RBF network. An important issue still is the set used for training, the number of epochs used and the optimal batch-size. In the problem presented here, local tables were used to generate prediction vectors for objects. In addition, the local tables contain only fragmented data, and to train the network we need objects that have values for all attributes that are present in the local tables (in order to designate vectors for all tables). So, to train the RBF network, a stratified 10-fold cross-validation method on the test set was used. That is, the test set was divided into 10 folds with equal number of objects and proportional shares of decision classes. At

each iteration, the RBF network was trained using 9 folds (to be very accurate – prediction vectors generated for test objects from these 9 folds), and the quality of classification of the model was evaluated on the last independent fold. This procedure was repeated three times and the results was averaged to determine the classification error level of the model. To determine the number of epochs and batch-size, different values were experimented to determine the optimal values for each dataset. After numerous trials, the optimal (epoch, batch-size) for Vehicle Silhouettes, Landsat Satellite and Dry Bean datasets were (400, 200), (400, 500), (400,15) respectively. RBF networks trains considerable fast so the whole process was fairly quick. A big advantage of the RBF networks was how interpretable the results obtained from it were. This was achieved thanks to properly selected/trained centers and their reduced influence in the case of large distances.

3 Datasets and results

The system proposed in the paper for the classification of objects in images based on characteristics stored in fragmented form – a set of local decision tables – was tested on three datasets. These datasets in their original form were retrieved from the UC Irvine Machine Learning Repository:

- Vehicle Silhouettes: eighteen quantitative conditional attributes, four decision classes, 846 objects – 592 training, 254 test set [17]. The goal of the data was to classify a given silhouette as one of four vehicle types, using a set of characteristics extracted from the silhouette. The vehicle can be viewed from one of many different angles. The images were acquired by a camera looking down at the vehicle model from a fixed elevation angle. The vehicles were rotated and their orientation angle was measured using a radial grid placed under the vehicle.
- Landsat Satellite: thirty-six quantitative conditional attributes, six decision classes, 6435 objects – 4435 training, 1000 test set [6]. Multispectral pixel values in a 3×3 neighborhood in a satellite image and a classification associated with the central pixel in each neighborhood. Each row of data corresponds to a neighborhood of pixels in a 3×3 square completely contained within an 82×100 sub-area. Each row contains the pixel values in the four spectral bands of each of the 9 pixels in the 3×3 neighborhood and a number indicating the classification label (earth type: red soil, cotton crop, grey soil, damp grey soil, soil with vegetation stubble, very damp grey soil) of the center pixel.
- Dry Bean: seventeen quantitative conditional attributes, seven decision classes, 13611 objects – 9527 training, 4084 test set [9]. The goal of the data was to classify type of beans based on the characteristics obtained from the image. Images of 7 different registered dry beans were taken with a high-resolution camera. Bean images obtained by computer vision systems were subjected to segmentation and feature extraction stages, and a total of 16 features – 12 dimensions and 4 shape forms were obtained from the grains.

Each of the datasets were originally available as a single decision table. However, the proposed system explored the possibilities of classification based on fragmentary data. Therefore, the data was preprocessed – randomly dispersed. Different numbers of local tables were considered during dispersion. The data was divided into $3, 5, 7, 9$ and 11 local tables. Each table contained a reduced set of conditional attributes and all objects from the original table. Some attributes were common between local tables. It should also be noted that, the more local tables, the fewer attributes in each local table. The data was imbalanced – the number of objects in individual decision classes, both in the training and test sets, varied strongly. The specific cardinality are presented in Figure 1. Two variants for each of the datasets were considered in the study. Experiments were performed both on dispersed imbalanced data and on data that had been modified by applying one of the known methods for imbalanced data. The Synthetic Minority Over-sampling Technique (SMOTE) method was used in the paper [4]. This is an over-sampling method which adds artificially created objects to a dataset. The objects were created based on randomly selected minority class objects. For this purpose, the $k-$nearest neighbors algorithm was used. On the line connecting the selected object with its closest neighbors, a new object from the minority class was created. The implementation of this algorithm available in WEKA [11] software was used. Each local table was balanced separately. Each decision class except the most numerous one, was changed using SMOTE method in such a way that all decision classes had the same number of objects after balancing. Thus, in the end, we obtained 30 dispersed datasets: Vehicle Silhouettes,

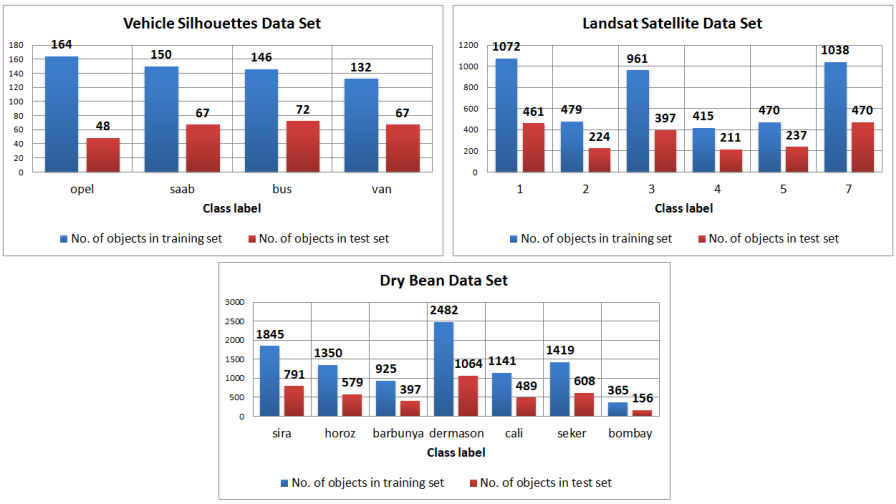


Fig. 1. Imbalance of data – cardinality of decision classes in training and test sets.

Vehicle Silhouettes balanced, Landsat Satellite, Landsat Satellite balanced, Dry

Bean, Dry Bean balanced; and for each dataset five versions of the dispersion: $3, 5, 7, 9, 11$ local tables. The quality of classification was evaluated based on the test set. A classification accuracy measure (acc) was used for this purpose. That is, a fraction of the total number of objects in the test set that were classified correctly. As was mentioned before, a 10-fold cross-validation was used on the test set, i.e., the neural network was trained 10 times with 9 folds and tested on one remaining fold. In addition, each test was performed three times to ensure that the results were reliable and not distorted by the influence of randomness. The results for the neural network approach that are given below is the average of the obtained results.

Three approaches were tested and the results are presented below. The two tested approaches use the RBF networks described above. The first RBF approach used a centers training stage by means of back-propagating the error up to the centers during training whiles the second RBF approach used only the Lloyd's $k-$means algorithm to determine the centers. As a baseline approach, the approach proposed in [10] was used. This ensemble of classifiers method consists of creating three base classifiers: $k-$nearest neighbors, decision tree and naive bayes classifier (KNN, DT, NB) based on each local table. The approach was implemented in python programming language using implementations available in the sklearn library. Based on each local table, the three classifiers were built. The final decision was made using soft voting for all classifiers. Different parameter values were tested for approaches based on RBF networks. For the generation of prediction vectors using the $k-$nearest neighbors classifier, $k \in \{1, 5, 10\}$ parameters were studied. Due to the limited space of this paper, only the results for the optimal k value is presented. Different numbers of neurons in the hidden layer were tested for RBF neural networks. The following values $\{0.25, 0.5, 0.75, 1, 1.5, 1.75, 2, 2.5, 2.75, 3, 3.5, 3.75, 4, 4.5, 4.75, 5\}\times$ the number of neurons in the input layer were tested. The results obtained for the RBF network with a centers training stage is presented in Table 1. The results obtained for the RBF network with the Lloyd's algorithm is shown in Tables 2. The best result for each dispersed dataset (each line) is shown in bold. As mentioned early on, different values of the parameter k were analyzed $k \in \{1, 5, 10\}$. In the tables, only the results obtained for $k = 5$ are presented because this value was optimal – in most cases for this value the best results were obtained. The results obtained for the ensemble of classifiers approach is given in Table 3.

Let us begin the analysis of the proposed RBF network with a centers training stage approach and RBF network with the Lloyd's algorithm by comparing the complexity of the neural nets for which optimal results were generated. As can be seen, significantly lower network complexity is sufficient to achieve the best results for the proposed approach. Figure 2 gives a comparison of the minimum number of neurons in the hidden layer sufficient to achieve the optimal result. The reduction in network's complexity using the proposed approach is significant compared to using the Lloyd's algorithm.

Now, we compare the results of classification accuracy obtained using the three analyzed approaches. Table 3 summarizes all results – the best results ob-

Table 1. Results of classification accuracy *acc* for the RBF network with a centers training stage. Designation I is used for the number of neurons in the input layer.

Dataset	No.	No. of neurons in hidden layer															
optimal k parameter	tables	0.25×I	0.5×I	0.75×I	1×I	1.5×I	1.75×I	2×I	2.5×I	2.75×I	3×I	3.5×I	3.75×I	4×I	4.5×I	4.75×I	5×I
Vehicle	3	0.483	0.608	0.642	0.661	0.702	0.719	0.725	0.735	0.734	0.759	0.756	0.763	0.752	**0.769**	0.762	0.765
imbalanced	5	0.568	0.651	0.667	0.683	0.716	0.724	0.733	0.75	0.754	0.758	0.754	0.743	0.75	0.749	0.749	**0.762**
k=5	7	0.567	0.639	0.671	0.668	0.685	0.702	**0.709**	0.7	0.701	0.694	0.685	0.682	0.69	0.68	0.682	0.684
	9	0.634	0.664	0.7	0.721	**0.73**	0.707	0.706	0.709	0.691	0.7	0.681	0.672	0.671	0.65	0.643	0.637
	11	0.6	0.654	0.69	0.707	0.711	0.702	0.707	0.705	**0.718**	0.703	0.714	0.714	0.707	0.712	0.707	0.702
Vehicle	3	0.721	0.735	0.741	0.735	0.73	0.744	0.752	0.752	0.748	0.759	0.756	0.744	0.76	0.755	0.755	**0.764**
balanced	5	0.709	0.72	0.755	0.757	0.754	**0.759**	0.749	0.745	0.755	0.745	0.745	0.734	0.73	0.725	0.719	0.724
k=5	7	0.726	0.727	**0.734**	0.714	0.722	0.706	0.707	0.7	0.684	0.684	0.667	0.658	0.653	0.629	0.625	0.62
	9	0.723	**0.732**	0.718	0.701	0.68	0.685	0.681	0.672	0.654	0.637	0.626	0.628	0.606	0.609	0.593	0.57
	11	**0.727**	**0.727**	0.72	0.722	0.702	0.698	0.693	0.668	0.668	0.659	0.647	0.631	0.597	0.593	0.58	0.588
Satellite	3	0.788	0.834	0.849	0.854	0.874	0.88	0.885	0.891	0.892	0.896	0.894	0.896	0.896	**0.897**	0.894	0.896
imbalanced	5	0.827	0.845	0.855	0.866	0.879	0.883	0.885	0.887	0.888	0.888	0.886	0.891	0.888	0.886	**0.892**	**0.892**
k=5	7	0.835	0.855	0.87	0.877	0.884	0.883	0.886	**0.887**	0.884	0.883	0.884	**0.887**	0.884	0.883	0.882	0.882
	9	0.845	0.857	0.87	0.877	0.884	**0.889**	0.883	0.886	0.883	0.885	0.883	0.88	0.882	0.882	0.881	0.882
	11	0.843	0.856	0.864	0.87	0.875	0.877	0.877	0.874	**0.878**	0.874	0.874	0.876	0.872	0.87	0.865	0.868
Satellite	3	0.659	0.775	0.813	0.832	0.852	0.856	0.858	0.864	0.864	0.867	0.867	0.868	0.872	0.873	**0.875**	0.874
balanced	5	0.753	0.808	0.839	0.849	0.85	0.855	0.854	0.858	0.858	0.863	0.862	0.864	**0.866**	0.865	**0.866**	**0.866**
k=5	7	0.78	0.834	0.852	0.851	0.85	0.856	0.859	0.861	**0.862**	0.861	0.861	0.86	0.861	0.86	0.859	0.859
	9	0.801	0.843	0.847	0.849	0.857	0.859	0.861	0.86	0.859	0.861	**0.863**	0.86	0.86	0.861	0.855	0.855
	11	0.815	0.843	0.846	0.845	0.851	0.851	0.855	0.859	**0.861**	0.859	0.857	0.854	0.85	0.849	0.848	0.846
Dry Bean	3	0.786	0.857	0.883	0.895	0.907	0.907	0.911	0.914	0.916	0.916	0.917	0.916	0.918	0.919	**0.92**	**0.92**
imbalanced	5	0.851	0.893	0.904	0.912	0.915	0.916	0.917	**0.918**	0.917	**0.918**	0.917	**0.918**	0.917	0.917	0.917	0.917
k=5	7	0.871	0.903	0.913	0.914	0.916	0.916	**0.917**	**0.917**	0.916	**0.917**	**0.917**	**0.917**	**0.917**	0.916	0.916	0.915
	9	0.881	0.905	0.912	0.915	0.917	0.917	0.917	0.918	**0.919**	0.918	0.918	0.918	0.917	0.918	0.918	0.918
	11	0.888	0.908	0.912	0.915	**0.916**	0.914	**0.916**	0.915	**0.916**	0.915	0.915	**0.916**	**0.916**	0.914	0.914	0.915
Dry Bean	3	0.789	0.855	0.882	0.892	0.902	0.908	0.91	0.913	0.916	0.916	**0.918**	**0.918**	**0.918**	**0.918**	0.917	**0.918**
balanced	5	0.852	0.892	0.905	0.91	0.915	0.916	**0.918**	**0.918**	0.917	**0.918**	0.917	0.916	0.917	0.916	0.916	0.917
k=5	7	0.865	0.901	0.91	0.914	**0.917**	**0.917**	0.915	0.916	0.916	**0.917**	0.916	0.916	**0.917**	**0.917**	0.914	0.915
	9	0.882	0.906	0.911	0.916	0.918	**0.919**	0.918	0.918	**0.919**	0.917	0.918	0.917	0.918	0.915	0.917	0.917
	11	0.886	0.906	0.911	0.915	0.917	**0.918**	0.916	0.916	0.917	0.916	0.917	0.917	0.916	0.914	0.915	0.916

Table 2. Results of classification accuracy *acc* for the RBF network with the Lloyd's algorithm. Designation I is used for the number of neurons in the input layer.

Data set	No.	No. of neurons in hidden layer															
optimal k parameter	tables	0.25×I	0.5×I	0.75×I	1×I	1.5×I	1.75×I	2×I	2.5×I	2.75×I	3×I	3.5×I	3.75×I	4×I	4.5×I	4.75×I	5×I
Vehicle	3	0.268	0.351	0.427	0.476	0.54	0.55	0.571	0.588	0.613	0.629	0.631	0.639	0.642	0.641	0.65	**0.656**
imbalanced	5	0.289	0.429	0.487	0.529	0.57	0.574	0.612	0.625	0.638	0.631	0.657	0.657	0.671	0.67	0.688	**0.692**
k=5	7	0.341	0.424	0.479	0.523	0.566	0.597	0.603	0.633	0.638	0.638	0.656	0.661	0.669	0.671	0.672	**0.676**
	9	0.363	0.496	0.545	0.582	0.616	0.623	0.637	0.659	0.655	0.665	0.686	0.688	0.716	0.719	0.719	**0.727**
	11	0.359	0.469	0.537	0.575	0.601	0.61	0.623	0.651	0.647	0.659	0.674	0.683	0.687	0.687	0.686	**0.698**
Vehicle	3	0.556	0.692	0.689	0.691	0.722	0.722	0.742	0.738	0.746	0.731	0.741	0.746	0.75	0.751	0.75	**0.761**
balanced	5	0.509	0.618	0.643	0.659	0.692	0.719	0.71	0.718	0.731	0.735	0.744	**0.757**	0.751	0.753	0.748	0.747
k=5	7	0.683	0.695	0.702	0.71	0.735	0.717	0.739	0.731	0.733	0.725	0.731	**0.741**	0.739	0.734	0.738	0.738
	9	0.503	0.671	0.674	0.687	0.707	0.709	0.717	0.72	0.721	**0.729**	0.728	0.72	0.724	0.725	0.725	0.72
	11	0.561	0.611	0.637	0.691	0.71	0.715	0.713	**0.722**	**0.722**	0.715	0.705	0.709	0.701	0.701	0.694	0.694
Satellite	3	0.729	0.789	0.817	0.823	0.83	0.833	0.834	0.841	0.846	0.846	0.85	0.848	0.851	0.851	0.852	**0.853**
imbalanced	5	0.759	0.809	0.828	0.83	0.838	0.84	0.845	0.849	0.85	0.852	0.852	0.852	0.855	0.855	**0.859**	**0.859**
k=5	7	0.773	0.825	0.834	0.833	0.841	0.846	0.848	0.851	0.853	0.853	0.854	0.858	0.858	0.859	0.859	**0.863**
	9	0.814	0.832	0.832	0.839	0.841	0.845	0.843	0.849	0.849	0.851	0.857	0.857	0.859	0.861	0.861	**0.862**
	11	0.81	0.831	0.841	0.841	0.844	0.847	0.848	0.85	0.851	0.851	0.854	0.855	0.856	0.857	**0.86**	0.859
Satellite	3	0.56	0.674	0.72	0.757	0.791	0.804	0.81	0.818	0.825	0.829	0.837	0.84	0.843	0.846	0.848	**0.849**
balanced	5	0.629	0.714	0.758	0.787	0.815	0.819	0.822	0.839	0.841	0.842	0.843	0.845	0.848	**0.85**	0.847	0.847
k=5	7	0.644	0.75	0.799	0.809	0.823	0.829	0.838	0.843	0.843	0.843	0.845	0.846	0.849	0.85	**0.851**	0.85
	9	0.669	0.774	0.8	0.818	0.828	0.832	0.843	0.843	0.844	0.841	0.846	**0.847**	0.846	0.843	0.844	0.845
	11	0.714	0.789	0.814	0.82	0.834	0.837	0.837	0.841	0.844	0.846	0.849	0.851	0.848	**0.851**	0.85	0.849
Dry Bean	3	0.591	0.777	0.828	0.842	0.864	0.869	0.871	0.882	0.886	0.887	0.891	0.892	0.895	0.897	**0.9**	0.898
imbalanced	5	0.762	0.833	0.858	0.874	0.89	0.893	0.894	0.897	0.899	0.9	0.902	0.904	0.905	0.905	0.904	**0.906**
k=5	7	0.796	0.847	0.873	0.883	0.893	0.896	0.9	0.901	0.904	0.903	0.905	0.905	0.906	0.907	0.908	**0.909**
	9	0.817	0.856	0.877	0.885	0.892	0.894	0.897	0.899	0.9	0.901	0.901	0.901	0.904	0.905	**0.906**	**0.906**
	11	0.827	0.867	0.886	0.891	0.898	0.9	0.9	0.901	0.901	0.902	0.901	0.902	0.902	0.903	0.904	**0.905**
Dry Bean	3	0.58	0.774	0.818	0.841	0.859	0.873	0.876	0.884	0.885	0.889	0.891	0.892	0.893	0.893	**0.896**	0.894
balanced	5	0.766	0.835	0.854	0.87	0.886	0.891	0.892	0.895	0.898	0.9	0.899	0.903	0.902	0.901	0.903	**0.905**
k=5	7	0.798	0.852	0.869	0.881	0.891	0.894	0.898	0.9	0.901	0.901	0.902	0.905	0.906	0.907	0.908	**0.91**
	9	0.817	0.859	0.873	0.885	0.894	0.896	0.897	0.9	0.9	0.901	0.902	0.903	0.905	0.906	**0.907**	0.905
	11	0.828	0.868	0.882	0.889	0.896	0.897	0.9	0.901	0.902	0.902	0.901	0.902	0.903	**0.904**	**0.904**	**0.904**

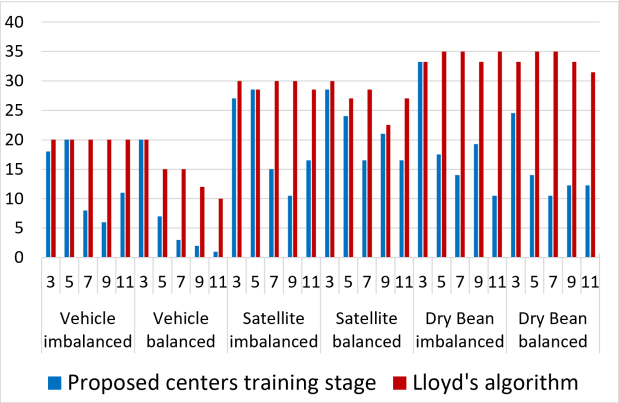


Fig. 2. The minimum number of neurons in the hidden layer sufficient to achieve the optimal result – comparison of RBF networks with a centers training stage approach and with the Lloyd's algorithm.

tained for the proposed approach, the best results using the RBF networks with the Lloyd's algorithm and the results for the ensemble of classifiers approach. The best result for each dispersed dataset (each line) is shown in bold. As can be seen, in all cases (except one) the best results were generated using the proposed approach. Statistical tests were performed in order to confirm significant differences in the obtained results *acc*. The received classification accuracy were divided into three dependent data samples, results from Table 3. The Friedman test was used to detect differences in multiple test samples. There was a statistically significant difference in the results obtained for the three different approaches being considered, $\chi^2(29, 2) = 39.467, p = 0.000001$. Additionally, comparative box-whiskers charts for the results with three approaches were created (Fig. 3). As can be observed, the values of the classification accuracy for the proposed approach – RBF network with a centers training stage is the best (much better than the others approaches). In the next step, the Wilcoxon each-pair test was used. This test confirmed that the differences in the classification accuracy were significant between the RBF network with a centers training stage and both the RBF network with the Lloyd's algorithm and the ensemble of classifiers approach. There in no statistically significant difference in the classification accuracy between the RBF network with the Lloyd's algorithm and the ensemble of classifiers approach.

For the proposed approach, a comparison of the results obtained by using balanced and imbalanced datasets were also made. Figure 4 shows the comparison of the results in a bar chart and box-whiskers charts. As can be seen for the Dry Bean set, balancing the dataset had no effect on the results. For the Satellite set, better results were obtained for the imbalanced dataset. On the other hand, for the Vehicle dataset, for a smaller number of local tables (3 and 5 tables) we got

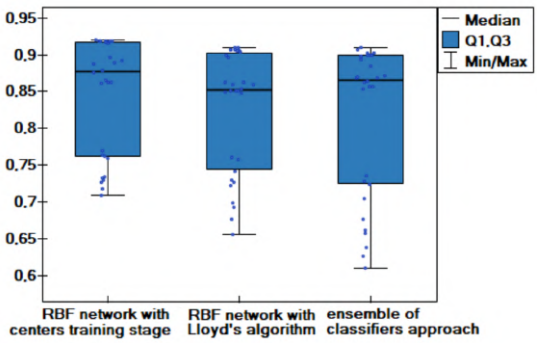


Fig. 3. Comparison of the results obtained for the three approaches: the RBF network with the centers training stage, the RBF network with the Lloyd's algorithm and the ensemble of classifiers approach.

better results for the imbalanced data. Also, the box-whiskers charts confirmed that there was no difference between the results for balanced and imbalanced datasets. The Wilcoxon test confirmed that there was no statistically significant difference in the average classification accuracy obtained for balanced and imbalanced sets for the proposed approach. Thus, it can be concluded that the proposed approach performs very well for imbalanced data.

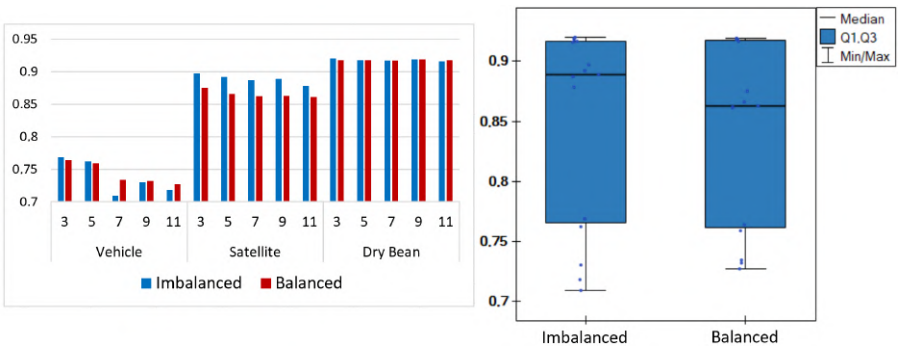


Fig. 4. Comparison of the results obtained for the RBF network with a centers training stage and imbalanced versus balanced datasets.

For the proposed approach, a comparison of the results obtained for different versions of dispersion was made. Figure 5 shows the comparison of the results in a bar chart and box-whiskers charts. As can be seen for the Dry Bean dataset, degree of dispersion of the dataset had no effect on the results. For the Satellite and the Vehicle datasets, better results were obtained for a smaller number of

Table 3. Comparison of classification accuracy *acc* obtained for approaches: the RBF network with a centers training stage, the RBF network with the Lloyd's algorithm and the ensemble of classifiers approach from paper [10].

Data set	No. tables	RBF network with a centers training stage	RBF network with Lloyd's algorithm	ensemble of classifiers from paper [10]
Vehicle imbalanced k=5	3	**0.769**	0.656	0.657
	5	**0.762**	0.692	0.638
	7	**0.709**	0.676	0.626
	9	**0.73**	0.727	0.661
	11	**0.718**	0.698	0.61
Vehicle balanced k=5	3	**0.764**	0.761	0.728
	5	**0.759**	0.757	0.724
	7	0.734	**0.741**	0.736
	9	**0.732**	0.729	0.705
	11	**0.727**	0.722	0.677
Satellite imbalanced k=5	3	**0.897**	0.853	0.885
	5	**0.892**	0.859	0.872
	7	**0.887**	0.863	0.868
	9	**0.889**	0.862	0.868
	11	**0.878**	0.86	0.863
Satellite balanced k=5	3	**0.875**	0.849	0.87
	5	**0.866**	0.85	0.864
	7	**0.862**	0.851	0.856
	9	**0.863**	0.847	0.856
	11	**0.861**	0.851	0.854
Dry Bean imbalanced k=5	3	**0.92**	0.9	0.906
	5	**0.918**	0.906	0.902
	7	**0.917**	0.909	0.899
	9	**0.919**	0.906	0.894
	11	**0.916**	0.905	0.9
Dry Bean balanced k=5	3	**0.918**	0.896	0.909
	5	**0.918**	0.905	0.899
	7	**0.917**	0.91	0.9
	9	**0.919**	0.907	0.898
	11	**0.918**	0.904	0.903

local tables – a smaller degree of dispersion, but the differences were not large. The Wilcoxon test confirmed that there was statistically significant differences in the average classification accuracy only between pairs of dispersion: 3 and 7 local tables; 5 and 7 local tables; 9 and 11 local tables. The conclusion of this comparison is that the proposed approach handles both small and large data dispersion quite well which is a very important property because often in real situations we have to deal with large dispersion – many units providing independent datasets. The proposed method requires optimization of several parameters in both the k-nearest neighbors classifier and the neural network, which can be considered a drawback of the method.

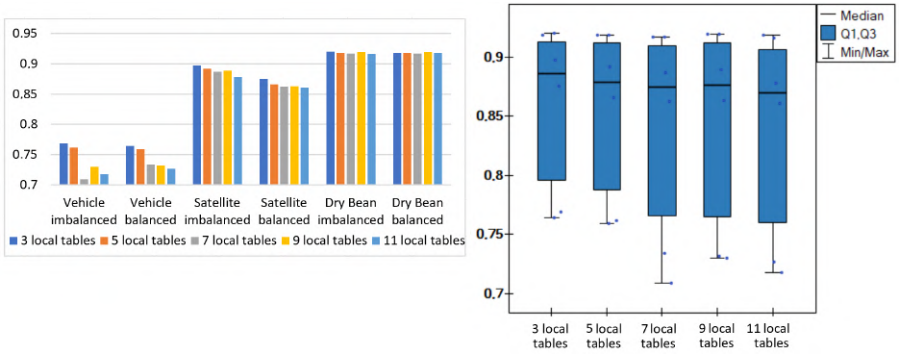


Fig. 5. Comparison of the results obtained for the RBF network with a centers training stage and different dispersion versions of datasets – 3, 5, 7, 9 and 11 local tables.

4 Conclusion

The study concerned analysis of the classification problem of an object presented in an image based on the characteristics of the object stored in a fragmentary form – a set of local decision tables. For this purpose, a RBF network model with a centers training stage was proposed. The paper shows that this approach gives much better results than the RBF network model with the Lloyd's algorithm. In addition, the network's structure is much simpler for the proposed approach. Moreover, the proposed approach gives better results than the ensemble of classifiers approach. It was also shown that the proposed model copes very well with imbalanced datasets and that the degree of dispersion does not have a large impact on classification accuracy. In future works, it is planned to use neural networks to define predictions based on local tables. The possibility of using a global learning stage after building the RBF network that combines predictions is also being considered. This stage would be implemented by using some artificially generated data.

References

1. Basha, S. S., Dubey, S. R., Pulabaigari, V., Mukherjee, S.: Impact of fully connected layers on performance of convolutional neural networks for image classification. Neurocomputing, 378, 112–119, (2020)
2. Caggiano, A., Zhang, J., Alfieri, V., Caiazzo, F., Gao, R., Teti, R.: Machine learning-based image processing for on-line defect recognition in additive manufacturing. CIRP annals, 68(1), 451–454, (2019)
3. Chaki, J., Dey, N.: Fragmented handwritten digit recognition using grading scheme and fuzzy rules. Sādhanā, 45(1), 1–23, (2020)
4. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res., 16:321–357, 2002.

5. Chen, J., Liu, G., Chen, X.: AnimeGAN: a novel lightweight GAN for photo animation. In International symposium on intelligence computation and applications 242–256. Springer, Singapore, (2020)
6. Dua, D. and Graff, C.: UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, (2019)
7. Fornasier, M., Toniolo, D.: Fast, robust and efficient 2D pattern recognition for re-assembling fragmented images. Pattern Recognition, 38(11), 2074–2087, (2005)
8. Giuseppe, A., Falchi, F.: k-NN based image classification relying on local feature similarity. International Conference on SImilarity Search and Applications, SISAP '10, Istanbul, Turkey.
9. Koklu, M., Ozkan, I. A.: Multiclass classification of dry beans using computer vision and machine learning techniques. Computers and Electronics in Agriculture, 174, 105507, (2020)
10. Kurian, R. A., Lakshmi, K.: An ensemble classifier for the prediction of heart disease. International Journal of Scientific Research in Computer Science, 3 , 25–31, (2018)
11. Markov, Z., Russell, I.: An introduction to the WEKA data mining system. SIGCSE Bull. 38, 3 (2006), 367–368. https://doi.org/10.1145/1140123.1140127
12. Mozgovoy, D. K., Hnatushenko, V. V., Vasyliev, V. V.: Automated recognition of vegetation and water bodies on the territory of megacities in satellite images of visible and IR bands. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 4(3), 167-172, (2018)
13. Przybyła-Kasperek, M., Wakulicz-Deja, A.: Global decision-making system with dynamically generated clusters, Inform. Sci. 270, 172–191 (2014)
14. Ptucha, R., Such, F. P., Pillai, S., Brockler, F., Singh, V., Hutkowski, P.: Intelligent character recognition using fully convolutional neural networks. Pattern recognition, 88, 604–613, (2019)
15. Santos, I., Castro, L., Rodriguez-Fernandez, N., Torrente-Patino, A., Carballal, A.: Artificial neural networks and deep learning in the visual arts: A review. Neural Computing and Applications, 33(1), 121–157, (2021)
16. Shelepin, Y. E., Chikhman, V. N., Foreman, N.: Analysis of the studies of the perception of fragmented images: global description and perception using local features. Neuroscience and behavioral physiology, 39(6), 569–580, (2009)
17. Siebert, J. P.: Vehicle Recognition Using Rule Based Methods, Turing Institute Research Memorandum TIRM-87-0.18, March 1987.
18. Sochor, J., Špaňhel, J., Herout, A.: Boxcars: Improving fine-grained recognition of vehicles using 3-d bounding boxes in traffic surveillance. IEEE Transactions on Intelligent Transportation Systems, 20(1), 97–108, (2018)
19. Traore, B. B., Kamsu-Foguem, B., Tangara, F.: Deep convolution neural network for image recognition. Ecological Informatics, 48, 257–268, (2018)
20. Wan, S., Goudos, S.: Faster R-CNN for multi-class fruit detection using a robotic vision system. Computer Networks, 168, 107036, (2020)
21. Yadav, S. S., Jadhav, S. M.: Deep convolutional neural network based medical image classification for disease diagnosis. Journal of Big Data, 6(1), 1–18, (2019)
22. Yang, R., Sarkar, S.: Gesture recognition using hidden markov models from fragmented observations. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06) vol. 1, 766–773. IEEE, (2006)

Publication [P5]

Marfo K.F.,Przybyła-Kasperek M. Study on the Use of Artificially Generated Objects in the Process of Training MLP Neural Networks Based on Dispersed Data *Entropy*, 25(5), 2023.
URL: https://doi.org/10.3390/e25050703.
DOI: 10.3390/e25050703.
$\text{MEiN}_{2023} = 100$

Number of citations:

- according to Web of Science: 1
- according to Google Scholar: 1

Contributor	Description of main tasks
Kwabena Frimpong Marfo	- investigation - conceptualization and methodology - result analysis - implementation of proposed algorithm - manuscript: review and editing - visualization: creating all figures in manuscript
Małgorzata Przybyła-Kasperek	- conceptualization and methodology - result analysis - manuscript: original draft preparation - manuscript: review and editing - visualization: creating all figures in manuscript

entropy

MDPI

Article

Study on the Use of Artificially Generated Objects in the Process of Training MLP Neural Networks Based on Dispersed Data

Kwabena Frimpong Marfo and Małgorzata Przybyła-Kasperek *

Institute of Computer Science, University of Silesia, Będzińska 39, 41-200 Sosnowiec, Poland; kwabena.marfo@us.edu.pl

* Correspondence: malgorzata.przybyla-kasperek@us.edu.pl; Tel.: +48-32-269-17-56

Abstract: This study concerns dispersed data stored in independent local tables with different sets of attributes. The paper proposes a new method for training a single neural network—a multilayer perceptron based on dispersed data. The idea is to train local models that have identical structures based on local tables; however, due to different sets of conditional attributes present in local tables, it is necessary to generate some artificial objects to train local models. The paper presents a study on the use of varying parameter values in the proposed method of creating artificial objects to train local models. The paper presents an exhaustive comparison in terms of the number of artificial objects generated based on a single original object, the degree of data dispersion, data balancing, and different network structures—the number of neurons in the hidden layer. It was found that for data sets with a large number of objects, a smaller number of artificial objects is optimal. For smaller data sets, a greater number of artificial objects (three or four) produces better results. For large data sets, data balancing and the degree of dispersion have no significant impact on quality of classification. Rather, a greater number of neurons in the hidden layer produces better results (ranging from three to five times the number of neurons in the input layer).

Keywords: neural network; multilayer perceptron; artificial training objects; independent data sources; dispersed data

Citation: Marfo, K.F.; Przybyła-Kasperek, M. Study on the Use of Artificially Generated Objects in the Process of Training MLP Neural Networks Based on Dispersed Data. *Entropy* **2023**, *25*, 703. https://doi.org/10.3390/e25050703

Academic Editor: Friedhelm Schwenker

Received: 23 March 2023
Revised: 19 April 2023
Accepted: 21 April 2023
Published: 24 April 2023

1. Introduction

A major problem in the domain of solving problems using machine learning is the decentralization of data sets and the inconsistency of information stored in local independent bases. When data is collected independently by institutions such as banks, hospitals, and various types of mobile applications, one cannot expect the format of the data to be uniform and consistent. Rather, one should expect that different sets of attributes and different sets of objects are present in local tables. Additionally, inconsistencies in data very often occur. The research presented in this paper deals precisely with the issue of classification based on dispersed data. By dispersed data, we mean data that are collected in several decision tables that contain inconsistencies, have different sets of attributes, and objects with the possibility that some attributes and objects may be common among decision tables. In addition, it is almost impossible to identify which objects are common among decision tables since to do that would require the existence of some central identifier of objects, which more often than not does not exist or may not be accessible due to data protection reasons.

The two main approaches that can be used for dispersed data are ensemble of classifiers and federated learning. Ensemble learning is a general approach of creating local models independently based on local tables [1,2], after which a final prediction is generated based on the local models by applying some fusion method [3–5]. In this approach, there is no global model as such.

In federated learning, a global model is built which constitutes the main objective presented in [6,7]. In this approach, the main focus is on data protection and data privacy [8]. Here, models are created in local spaces and their parameters only are sent to a central server—local data are not exchanged or combined among local spaces. The local models are then aggregated and sent to the local spaces. Such a procedure is iterated until a convergence criterion is satisfied.

The approach proposed in this paper is quite different. The aim of the method is to build a global model but in a completely different way than in federated learning. Indeed, local models are built based on local tables which are later used to construct a global model; however, this procedure is not iterative. Creation of a global model is carried out by a one-time aggregation. In the final stage, the global model is trained with a stratified subset of the original test set for which the values on the full set of conditional attributes present in all local tables are defined.

In this study, neural networks—multilayer perceptrons (MLP)—are used as local models. For the aggregation of such local networks to be possible, all of them must have the same structure. Since there are different conditional attributes in each local table, obtaining the same input layer in all models is not trivial. It is necessary to artificially generate objects based on the original objects that are to be used to train the network. Such artificial objects must have defined values on the conditional attributes that are missing in the considered local table. The paper proposes a method for generating artificial objects and contains a study on the use of different parameter values in the proposed method of generating artificial objects. An exhaustive comparison in terms of the number of artificial objects generated based on a single original object, the degree of data dispersion, data balancing, and different network structures—the number of neurons in the hidden layer are presented. The main conclusions reached are as follows: it was found that for data sets with a large number of objects, a smaller number of artificial objects is optimal. For smaller data sets, a greater number of artificial objects (three or four) produces better results. For large data sets, data balancing and the degree of dispersion have no significant impact on the quality of classification. Rather, a greater number of neurons in the hidden layer produces better results (ranging from three to five times the number of neurons in the input layer).

The contribution of the paper are as follows:

- Proposing a method for generating artificial objects for training local MLP networks with identical structure;
- Comparison of the proposed method in relation to different number of artificially objects generated;
- Comparison of the proposed method in relation to different versions of data dispersion;
- Comparison of the proposed method in relation to different number of neurons in the hidden layer;
- Comparison of the proposed method for balanced and imbalanced versions of data sets.

Neural networks have been considered for dispersed data in various applications. The papers [9,10] considered neural networks as a model for aggregating prediction vectors generated by local classifiers. In the paper [11], neural networks were used in a federated learning approach. Neural networks were also used as base models in an ensemble of classifiers whose predictions were then aggregated by various fusion methods [12]. However, none of the approaches described above is similar to the one proposed in this study. The main difference lie in the non-iterative approach when building the global model in the proposed approach and the use of local tables with different sets of conditional attributes to train local networks with identical structures.

The paper is organized as follows. In Section 2, the proposed method for generating a global model is described. The section explains how to determine the structure of local models and how to prepare artificial objects for training local models. Then, the method of aggregating local models to the global model and the stage of training the global model are described. Section 3 addresses the data sets that were used and presents the conducted

experiments, comparisons, and discussion on obtained results. Section 4 is on conclusions and future research plans.

2. Materials and Methods

The main idea of the proposed model is to build a global model based on dispersed data—local tables with different sets of conditional attributes—in three stages:

- First stage: training local models, MLP neural networks based on local tables;
- Second stage: aggregation of local models to the global model. This stage is performed in a non-iterative way by a single calculation;
- Third stage: post-training the global model using a stratified subset of the original test set.

All three stages are described below in separate subsections.

2.1. First Stage—Training Local Models, MLP Neural Networks, Based on Local Tables

Formally, dispersed data is a set of decision tables that are collected independently by separate units. We assume that a set of decision tables—local tables $D_i = (U_i, A_i, d)\ i \in \{1, \ldots, n\}$ from one discipline—is available, where U_i is the universe comprising a set of objects; A_i is a set of conditional attributes; and d is a decision attribute. We assume that the sets of conditional attributes of local tables are quite different although it may rarely happen that a larger set of attributes is common between tables. More likely, the differences in attributes found in local tables are significant.

The local models that are used in this study are multilayer perceptron networks (MLP). Based on each local table, an MLP model is trained separately. The desired objective that all local models must have the same structure is not trivial since each local table has different conditional attributes, thus making the training process difficult. We propose that the input layer of local networks contains all the attributes that are present in all local tables—let us denote this set as $A = \bigcup_{i \in \{1,\ldots,n\}} A_i$. In addition, the hidden layer should contain the same number of neurons in all networks. The output layer will be same for all tables due to the identical decision attribute present in all local tables. In this study, we use only one hidden layer in the network.

Now, a problem arises when we seek to train such a network based on a single local table given that the table in question lacks conditional attributes (perhaps many) that are present in the input layer of the network. A method for generating artificial objects with supplemented values on missing conditional attributes is proposed. These values are imputed based on certain characteristics provided by other local tables in the dispersed data in which the missing attributes are present. In doing so, data protection is ensured because we do not exchange raw data but only certain values of statistical measures derived from the dispersed data.

Based on each original object from a local table, k artificial objects are generated as follows:

1. Let us consider an object x that belongs to a decision class v from a local table D_i.
2. We define a set of tuples as

$$\begin{aligned} METHODS &= (min, min), (min, mean) \cdots (max, median), (max, max) \\ &\in (min, mean, median, max) \times (min, mean, median, max) \end{aligned}$$

   For each missing attribute (attribute from the set $A \setminus A_i$) and each $method \in METHODS$, $method(0)$ is computed on the objects having the decision class v for all local tables in which the attribute is present. After, $method(1)$ is computed on the the resulting values from $method(0)$.
3. After step 2, there will be $|METHODS| = 16$ values for decision class v. k distinct values denoted by a_k are randomly selected from the 16 values, where k is the number of artificial objects that are to be generate.
4. From step 3, there will be k derived values for all the missing attributes of object x.

5. The final step is to duplicate object x, k times, and assign the a_k values to the missing attribute.

This process is carried out for all objects in a local table and executed separately for each local table.

A training set of artificially prepared objects as described above is then used to train the MLP network. The neural networks is implemented using the Keras library in Python. Different number of neurons in the hidden layer is experimented on—values ranging from 0.25 to 5 times the number of neurons from the input layer are tested. For the hidden layer, the ReLU (Rectified Linear Unit) activation function is used as it is the most popular activation function and gives very good results [13]. For the output layer, the Softmax activation function is used, which is recommended when we deal with a multi-class problem [14]. The neural network is trained by using a gradient descent method with an adaptive step size in the backpropagation method. The Adam optimizer [15] and the categorical cross-entropy loss function [16] are used in the study.

2.2. Second Stage—Aggregation of Local Models to the Global Model

The second stage consists of aggregation of local networks into a single global network. In the first stage, the local neural networks are prepared in such a way that aggregation is possible—all local networks have the same structure; thus, the global network will also have the same network structure. The weights in global model are determined based on the weighted average of the corresponding weights from the local models. However, due to the dispersed data stored in the local tables, not all local models are equally accurate, so the weighted average is employed to make the local model's influence on the construction of the global model depend on the accuracy of a given local model. The method used is inspired by the second weighting system used in the AdaBoost algorithm [17].

For each local model, a classification error is estimated based on its training set (containing artificial objects). Let us denote by e_i the classification error determined for the i-th local model $i \in \{1, \ldots, n\}$. Since local models are built based on a piece of data, their accuracy can be very different. It may sometimes happen that their classification error is above 0.5. In order not to eliminate such local models from the aggregation stage as they may contain important information on specific attributes that may have a positive impact in the global model, the min-max normalization is applied to the interval $[0, 0.5]$ of all errors $e_i, i \in \{1, \ldots, n\}$. After, the weights ω_i for each local neural network $i \in \{1, \ldots, n\}$ are adjusted according to the formula proposed in [17]:

$$\omega_i = ln(\frac{1 - e_i}{e_i}) \tag{1}$$

The initial weights of the global model between neural connections are then calculated based on the adjusted weights of all the local networks. More specifically, the weights of the global model are determined by the weighted average of adjusted weights $\omega_i, i \in \{1, \ldots, n\}$.

It should be noted that some attributes that appear more frequently in local tables may have been better trained in global model than others. Therefore, a MLP network created in this way does not always generate sufficiently good results. In the next stage, the quality of the network is improved.

2.3. Third Stage—Post-Training the Global Model Using a Small Training Set

In order to implement this step, it is necessary to have access to an independent set of training data which can be called a global training set. This means that each object in this set has values for all conditional attributes A from the dispersed data. This set cannot be generated from local tables since aggregation is not possible considering the assumptions about dispersed data mentioned earlier.

Such a global training set is extracted from the test set. The test set is divided into two equal parts in a stratified manner. One is used for the post-training stage and the other for testing. This procedure is repeated twice where each time a different half is used for

the post-training phase. In future studies, it is planned to generate such a global training set artificially.

3. Results

The experiments are conducted with data taken from the UC Irvine Machine Learning Repository. Three data sets are selected: Vehicle data [18], Landsat Satellite data [19], and Dry Bean data [20]. Each data set available in the repository is stored in a single table. These data sets are chosen for three reasons. To begin, these data sets are chosen because of the presence of multiple decision classes in the sets as the proposed method is tested for multi-class problems. Additionally, an important factor is the significant number of conditional attributes present in the data sets. The data are dispersed into local tables in the way where the conditional attributes are split. The aim is to test the approach where we have different conditional attributes in local tables. To achieve this, a large number of attributes is needed originally so that such dispersion can occur and a meaningful subset of these attributes can be present in each local table. Lastly, in this study, we focus on using numerical data—there are numerical, discrete, or continuous attributes in all data sets. Due to the large variation in the attributes in the Dry Bean data, the set is normalized.

The only possible way to evaluate the model for the considered dispersed data is the train-and-test method. This is because the data in the local tables contain only subsets of conditional attributes, while we assume that the test objects will already have specified values for all possible attributes present in the local tables. So, before the original data set is dispersed, it is divided into a training set (70% of objects) and a test set (30% of objects) in a stratified manner. Data characteristics are given in Table 1. The training data sets are then dispersed into local tables. Different degrees of dispersion are considered in order to check whether the method can cope with significant data dispersion. The creation of versions with $3, 5, 7, 9$, and 11 local tables based on the original training set are considered where all local tables contained only a subset of the original set of conditional attributes. In addition, different local tables had different sets of attributes; however, there is a possibility of individual attributes being present among some tables. The decision attribute is included in each of the tables. The full set of objects is also stored in each of the local tables but without identifiers. This reflects the real situation where one cannot identify the objects between local tables.

Table 1. Data set characteristics. Sign # denotes the number of objects in the set.

Data Set	# The Training Set	# The Test Set	# Conditional Attributes	# Decision Classes
Vehicle	592	254	18	4
Landsat Satellite	4435	1000	36	6
Dry Bean	9527	4084	16	7

All the data sets are heavily imbalanced Figure 1. To check whether the proposed method can handle imbalanced data, each data set is considered in two versions—the imbalanced version and the balanced version. The data are balanced with the use of the synthetic minority over-sampling technique (SMOTE) method [21]. The implementation of this algorithm, available in WEKA [22] software, is used. The balancing procedure is performed for each local table separately using only the locally available subset of attributes. All objects for each decision class are balanced in a way that after the implementation of this process, each decision class has an equal number of objects as the decision class with the most objects in the set. Thus, a total of thirty dispersed sets are analyzed: each of the three data sets is dispersed into 5 versions, each version is balanced to a total of $3 \times 5 \times 2$.

The quality of classification is evaluated based on the test set. The accuracy measure *acc* is analyzed. This is the defined as a fraction of correctly classified objects to all objects in the test set.

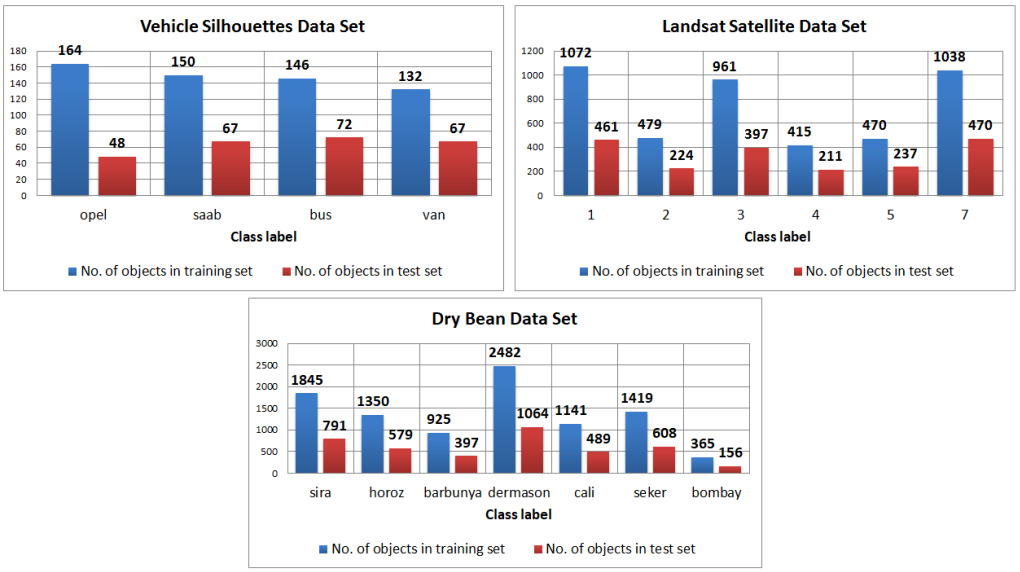


Figure 1. Imbalance of data—cardinality of decision classes in training and test sets.

The main goal of the experiments is to investigate how the number of objects artificially generated based on a single object from a local table affects the quality of classification. An additional purpose is to determine the guidelines that should be followed in determining such an optimal value depending on the characteristics of the data sets as well as to check the effect of the degree of dispersion on the obtained quality of classification. The different network structures and the impact of the number of neurons in the hidden layers on the quality of classification are also studied. Comparison analysis to determine whether the proposed approach performs equally well for balanced and imbalanced data is carried out. To meet these objectives above, the scheme of the experiments is as follows.

- Studying different number of artificial objects generated based on a single object from each local table. The number of artificial objects generated $k \in \{1, 2, 3, 4, 5\}$ are studied.
- Studying different levels of dispersion: 3, 5, 7, 9, 11 local tables.
- Studying different number of neurons in the hidden layer. The number is determined in proportion to the number of neurons in the input layer. The following values are tested: {0.25, 0.5, 0.75, 1, 1.5, 1.75, 2, 2.5, 2.75, 3, 3.5, 3.75, 4, 4.5, 4.75, 5} $\times$ the number of neurons in the input layer.
- Studying two versions for each data set—balanced and imbalanced versions.
- Studying an iterative approach modeled on federated learning in order to make comparisons with the proposed approach.

Comparison of experimental results is made in terms of:

- The quality of classification for different number of artificial objects generated;
- The quality of classification for different versions of dispersion;
- The quality of classification for different number of neurons in the hidden layer;
- The quality of classification for balanced and imbalanced version of data sets.

Tables A1–A6, presented in Appendix A, show the classification accuracy obtained for different versions of dispersion, different numbers of artificially generated objects, and different numbers of neurons in the hidden layer for Vehicle imbalanced, Vehicle balanced, Landsat Satellite imbalanced, Landsat Satellite balanced, Dry Bean imbalanced and Dry Bean balanced data sets. Each experiment is performed three times. The average of the three runs is given in the tables below. In each row of the tables, the best result is in a bold font. The following sections present an analysis of the results included in these tables from different perspectives. The last part presents a comparison with the approach modeled on federated learning.

3.1. Comparison of Quality of Classification for Different Numbers of Objects Artificially Generated

First, we compare the quality of classification using different number of artificially generated objects. Table 2 shows a comparison of the best results (those in a bold font in Tables A1–A6) obtained for different number of artificially generated objects. In the table, for each dispersed data set, the best result is shown in a bold font.

Table 2. Comparison of classification accuracy *acc* obtained for different number of artificially generated objects.

Data Set	No. of Tables	No. of Artificially Generated Objects				
		1	2	3	4	5
Vehicle imbalanced	3	0.724	0.688	**0.73**	0.702	0.693
	5	0.71	0.696	**0.728**	0.724	0.714
	7	0.698	0.699	**0.72**	0.719	0.713
	9	0.698	0.707	0.709	**0.727**	0.703
	11	0.694	0.71	0.673	**0.728**	0.696
Vehicle balanced	3	0.73	0.705	0.726	**0.735**	0.731
	5	0.738	**0.748**	0.722	0.738	0.726
	7	0.757	0.739	0.756	0.752	**0.759**
	9	0.743	0.718	0.736	0.747	**0.773**
	11	0.705	**0.726**	0.706	0.719	0.713
Landsat Satellite imbalanced	3	0.809	0.809	**0.813**	**0.813**	0.808
	5	0.815	0.809	**0.82**	0.811	0.813
	7	**0.814**	0.809	0.811	0.81	0.809
	9	0.805	**0.813**	0.808	0.806	0.809
	11	0.804	**0.815**	0.81	0.8	0.804
Landsat Satellite balanced	3	0.799	0.799	**0.803**	0.797	**0.803**
	5	0.799	0.797	**0.805**	0.801	0.8
	7	0.8	0.791	**0.801**	0.793	0.791
	9	0.794	0.793	0.795	0.793	**0.796**
	11	0.79	0.791	0.789	**0.798**	0.792
Dry Bean imbalanced	3	0.915	**0.917**	0.913	**0.917**	0.914
	5	0.913	**0.915**	0.912	0.914	0.913
	7	0.911	**0.915**	0.911	0.91	0.914
	9	0.912	**0.913**	0.91	0.911	**0.913**
	11	0.912	**0.914**	0.908	0.911	0.911
Dry Bean balanced	3	0.915	**0.918**	0.913	0.916	0.913
	5	0.912	**0.916**	0.913	0.913	0.913
	7	0.911	**0.915**	0.913	0.912	0.911
	9	**0.913**	**0.913**	0.907	0.911	0.911
	11	0.911	0.911	**0.913**	0.912	0.909

As can be seen, for different data sets, different numbers of artificially generated objects guarantee the best results. In the case of the Vehicle data set, it can only be said that the approach with one artificial object gives the worst results. In the case of the Dry Bean data set, definitely the use of two artificial objects generates the best results. For the Landsat Satellite data set, it is hard to define any of these types of relations.

Statistical tests are performed in order to check the importance in the differences in the obtained results *acc* for different number of objects artificially generated. The Friedman's test using all results from Table 2 is performed. Five dependent groups are analyzed ($\{1, 2, 3, 4, 5\}$ number of artificial objects). The test did not confirm that differences among the classification accuracy in these five groups are significant ($p = 0.672$). However, as can be seen from Table 2, the classification accuracy obtained for different data sets are from completely different ranges. Due to this discrepancy, it is difficult to prove the significance of the differences. Therefore, it was decided to separate the obtained results against the considered data sets. Thus, three sets (for Vehicle, for Landsat Satellite, and for Dry Bean)

each containing a ten-element sample are obtained. The Friedman's test confirmed the significance of the differences for the Dry Bean data set with $p = 0.003$. The Wilcoxon each-pair test confirmed the significant differences between the average accuracy values for the following pairs: Vehicle—2 and 4 artificial objects, $p = 0.01$; Landsat Satellite—1 and 3 artificial objects, $p = 0.03$; Dry Bean—2 and 1 artificial objects, $p = 0.008$, 2 and 3 artificial objects, $p = 0.006$, 2 and 4 artificial objects, $p = 0.008$, 2 and 5 artificial objects, $p = 0.004$.

Additionally, comparative box-plot charts for the values of the classification accuracy and different data sets are created (Figure 2). As can be observed, for the Dry Bean data set, the box-plot for the two artificial objects definitely stands out among the others. It can also be concluded that using a single artificial object never generates good results. Taking into account the results of the comparisons and the number of objects in the analyzed data sets, a general conclusion can be drawn. For data sets with a large number of objects (around 9000 objects), a smaller number of artificial objects such as two objects is optimal. For smaller data sets with up to a thousand objects, a greater number of artificial objects (three or four) produces better results. More specifically, the smaller the number of objects in the local tables, the more artificially generated objects should be used in the proposed approach.

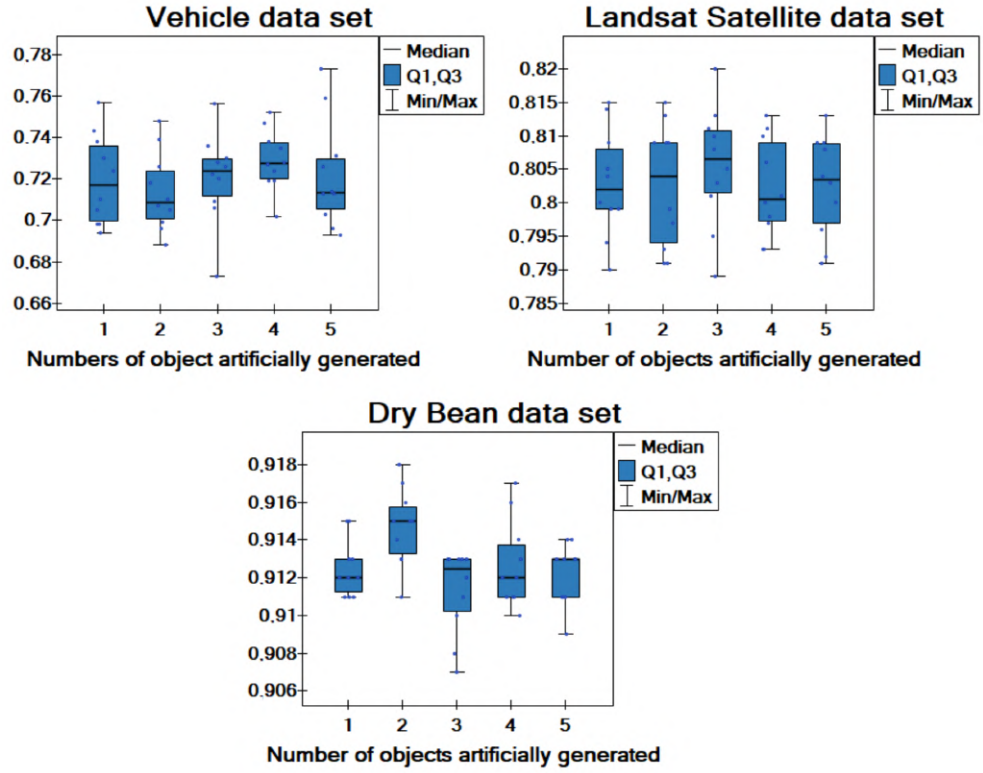


Figure 2. Box-plot chart with (median, the first quartile—Q1, the third quartile—Q3) the value of classification accuracy *acc* for the different numbers of objects artificially generated.

3.2. Comparison of Quality of Classification for Different Versions of Dispersion

We now compare the classification accuracy obtained for different versions of data dispersion. In Table 3 a comparison of the best results (those bolded in Tables A1–A6) obtained for different version of dispersion is presented. In the table, for data set, the best result is shown in a bold font.

As can be observed, in the case of Vehicle data set, the best results are obtained for medium data dispersion (7 local tables) or even large data dispersion (11 local tables). For this data set, the differences in results obtained for different versions of dispersion are the greatest compared to the other data sets. For Landsat Satellite and Dry Bean data sets, the smallest dispersion (3 local tables) gives better results. However, looking closely

at the results, we can observe that the absolute differences noted for these data sets are really small—at the third decimal place. So, we can conclude that for data sets with such a large number of objects, the differences recorded for different degrees of dispersion are really unremarkable.

Table 3. Comparison of classification accuracy *acc* obtained for different numbers of artificially generated objects.

Data Set	No. of Artificially Generated Objects	No. of Local Tables				
		3	**5**	**7**	**9**	**11**
Vehicle imbalanced	1	**0.724**	0.71	0.698	0.698	0.694
	2	0.688	0.696	0.699	0.707	**0.71**
	3	**0.73**	0.728	0.72	0.709	0.673
	4	0.702	0.724	0.719	0.727	**0.728**
	5	0.693	**0.714**	0.713	0.703	0.696
Vehicle balanced	1	0.73	0.738	**0.757**	0.743	0.705
	2	0.705	**0.748**	0.739	0.718	0.726
	3	0.726	0.722	**0.756**	0.736	0.706
	4	0.735	0.738	**0.752**	0.747	0.719
	5	0.731	0.726	0.759	**0.773**	0.713
Landsat Satellite imbalanced	1	0.809	**0.815**	0.814	0.805	0.804
	2	0.809	0.809	0.809	0.813	**0.815**
	3	0.813	**0.82**	0.811	0.808	0.81
	4	**0.813**	0.811	0.81	0.806	0.8
	5	0.808	**0.813**	0.809	0.809	0.804
Landsat Satellite balanced	1	0.799	0.799	**0.8**	0.794	0.79
	2	**0.799**	0.797	0.791	0.793	0.791
	3	0.803	**0.805**	0.801	0.795	0.789
	4	0.797	**0.801**	0.793	0.793	0.798
	5	**0.803**	0.8	0.791	0.796	0.792
Dry Bean imbalanced	1	**0.915**	0.913	0.911	0.912	0.912
	2	**0.917**	0.915	0.915	0.913	0.914
	3	**0.913**	0.912	0.911	0.91	0.908
	4	**0.917**	0.914	0.91	0.911	0.911
	5	**0.914**	0.913	**0.914**	0.913	0.911
Dry Bean balanced	1	**0.915**	0.912	0.911	0.913	0.911
	2	**0.918**	0.916	0.915	0.913	0.911
	3	**0.913**	**0.913**	**0.913**	0.907	**0.913**
	4	**0.916**	0.913	0.912	0.911	0.912
	5	**0.913**	**0.913**	0.911	0.911	0.909

Statistical tests are performed in order to confirm the importance in the differences in the obtained results *acc*. At first, the values of the classification accuracy in five dependent groups (3, 5, 7, 9, 11 local tables) are analyzed. The Friedman test confirmed a statistically significant difference in the results obtained for the five different version of dispersion being considered, $\chi^2(28, 4) = 26.608, p = 0.00003$. The Wilcoxon each-pair test confirmed the significant differences between the average accuracy values for all pairs with 11 local tables: 3 and 11 local tables $p = 0.007$, 5 and 11 local tables $p = 0.001$, 7 and 11 local tables $p = 0.004$, 9 and 11 local tables $p = 0.016$.

Additionally, a comparative box-plot chart for the values of the classification accuracy is created (Figure 3). Here, the distributions of the values obtained for different versions of dispersion are similar; thus, we can conclude that for sufficiently large data sets (5000 objects), the degree of dispersion does not have a huge impact on the obtained results. More specifically, the degree of dispersion has little effect on the quality of classification in the proposed approach.

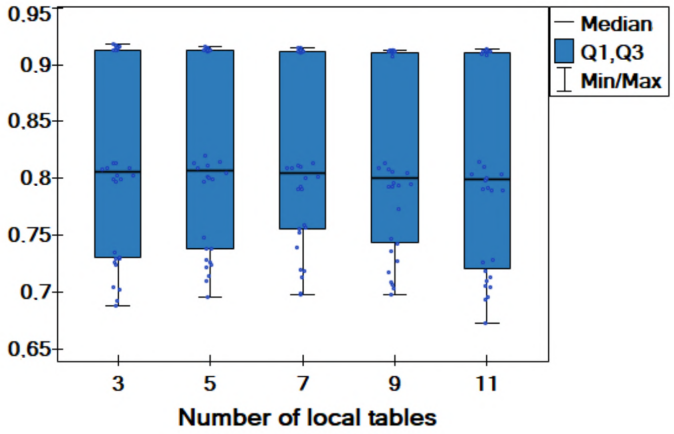


Figure 3. Box-plot chart with (median, the first quartile—Q1, the third quartile—Q3) the value of classification accuracy *acc* for different versions of dispersion.

3.3. Comparison of Quality of Classification for Different Numbers of Neurons in the Hidden Layer

In Tables A1–A6, which are presented earlier, all the results obtained for the different analyzed number of neurons in the hidden layer are given. The best obtained classification accuracies are also marked in those tables. It can be seen that these best results are generated by a higher number of neurons in the hidden layer. The optimal values are above $3\times$ the number of neurons in the input layer up to $5\times$ the number of neurons in the input layer. This propriety does not depend on the number of objects in data set—no matter how large the data set is, more neurons in the hidden layer gives better results. However, there is not one universal number of neurons in the hidden layer that is optimal for every data set.

In order to notice certain patterns for particular data sets, heat maps are created based on the results from Tables A1–A6 and shown in Figure 4. On the x-axis, the number of neurons in the hidden layer is presented, while the number of artificial objects generated and the version of the dispersion are shown on the y-axis. The color on the map is determined by the classification accuracy value. Definitely for the Dry Bean data set, the clearest pattern can be seen, which shows that increasing the number of neurons in the hidden layer clearly improves classification accuracy. Additionally, for the Vehicle data set, it can be seen that a higher number of neurons results in better quality. The least visible dependence is found in the heat map for the Landsat Satellite data set. Here, for a large number of neurons in the hidden layer, both very good classification quality and worse results were observed. More specifically, it depends on the data set whether the increased number of neurons in the hidden layer will improve the quality of classification, and this impact is very different and specific to the data set.

3.4. Comparison of Quality of Classification for Balanced and Imbalanced Versions of Data Set

We will now focus on comparing the results obtained for balanced and imbalanced data. In Table 4, a comparison of the best results (those in a bold font in Tables A1–A6) obtained for balanced and imbalanced versions of each dispersed data is presented. In the table, the best result is shown in a bold font for each dispersed data set.

Based on the results, it cannot be explicitly concluded that the proposed method gives better results for balanced only or imbalanced data only as it depends on the data set in question. For the Vehicle data set, better results are obtained with balanced data, while for the Landsat Satellite data set, better results are obtained with imbalanced data. In both cases, the Wilcoxon test for dependent samples confirmed the statistical significance of the differences with $p = 0.0001$. In contrast, for the Dry Bean data set, the results in both balanced and imbalanced versions are virtually the same. Here, the Wilcoxon test did not confirm the significance of the differences ($p = 0.523$).

Table 4. Comparison of classification accuracy *acc* obtained for imbalanced and balanced versions of data.

Data Set	No. of Tables	No. of Art. Objects	Imbalanced	Balanced	Data Set	Imbalanced	Balanced
Vehicle	3	1	0.724	**0.73**	Dry Bean	0.915	0.915
		2	0.688	**0.705**		0.917	**0.918**
		3	**0.73**	0.726		0.913	0.913
		4	0.702	**0.735**		**0.917**	0.916
		5	0.693	**0.731**		**0.914**	0.913
	5	1	0.71	**0.738**		**0.913**	0.912
		2	0.696	**0.748**		0.915	**0.916**
		3	**0.728**	0.722		0.912	**0.913**
		4	0.724	**0.738**		**0.914**	0.913
		5	0.714	**0.726**		0.913	0.913
	7	1	0.698	**0.757**		0.911	0.911
		2	0.699	**0.739**		0.915	0.915
		3	0.72	**0.756**		0.911	**0.913**
		4	0.719	**0.752**		0.91	**0.912**
		5	0.713	**0.759**		**0.914**	0.911
	9	1	0.698	**0.743**		0.912	**0.913**
		2	0.707	**0.718**		0.913	0.913
		3	0.709	**0.736**		**0.91**	0.907
		4	0.727	**0.747**		0.911	0.911
		5	0.703	**0.773**		**0.913**	0.911
	11	1	0.694	**0.705**		**0.912**	0.911
		2	0.71	**0.726**		**0.914**	0.911
		3	0.673	**0.706**		0.908	**0.913**
		4	**0.728**	0.719		0.911	**0.912**
		5	0.696	**0.713**		**0.911**	0.909
Landsat Satellite	3	1	**0.809**	0.799			
		2	**0.809**	0.799			
		3	**0.813**	0.803			
		4	**0.813**	0.797			
		5	**0.808**	0.803			
	5	1	**0.815**	0.799			
		2	**0.809**	0.797			
		3	**0.82**	0.805			
		4	**0.811**	0.801			
		5	**0.813**	0.8			
	7	1	**0.814**	0.8			
		2	**0.809**	0.791			
		3	**0.811**	0.801			
		4	**0.81**	0.793			
		5	**0.809**	0.791			
	9	1	**0.805**	0.794			
		2	**0.813**	0.793			
		3	**0.808**	0.795			
		4	**0.806**	0.793			
		5	**0.809**	0.796			
	11	1	**0.804**	0.79			
		2	**0.815**	0.791			
		3	**0.81**	0.789			
		4	**0.8**	0.798			
		5	**0.804**	0.792			

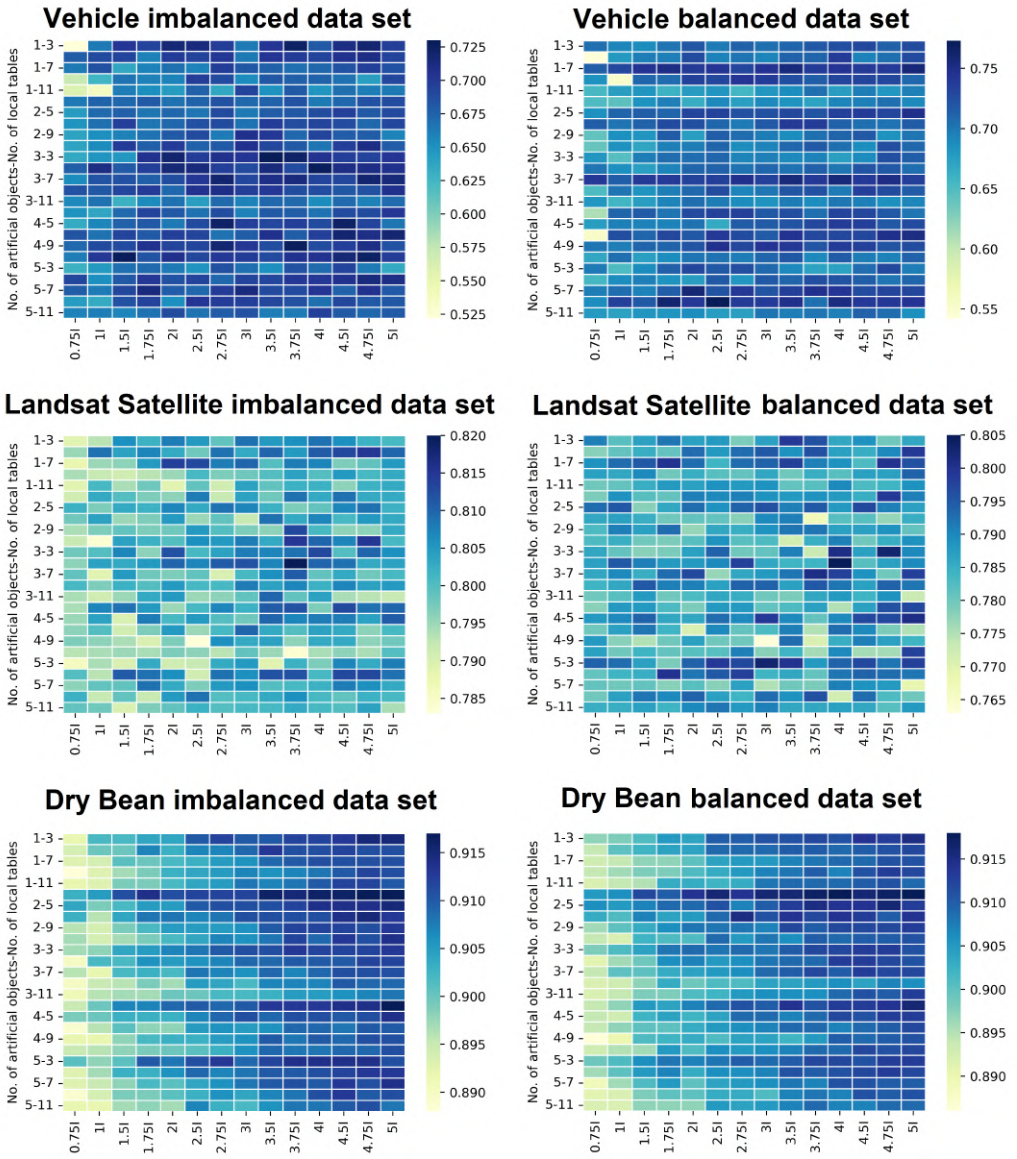


Figure 4. Heat maps on the accuracy levels of all data sets.

Comparative box-plot charts for the values of the classification accuracy in two groups of imbalanced and balanced data are created (Figure 5). The graphs confirm earlier conclusions; hence, it can be said that the proposed method handles balanced and imbalanced data comparably. In fact, the final result depends on the specifics of the data set. Determining the specific characteristics of the data sets that influenced the results requires further study. More specifically, it depends on the data set whether applying the SMOTE method for balancing the data set improves the quality of classification.

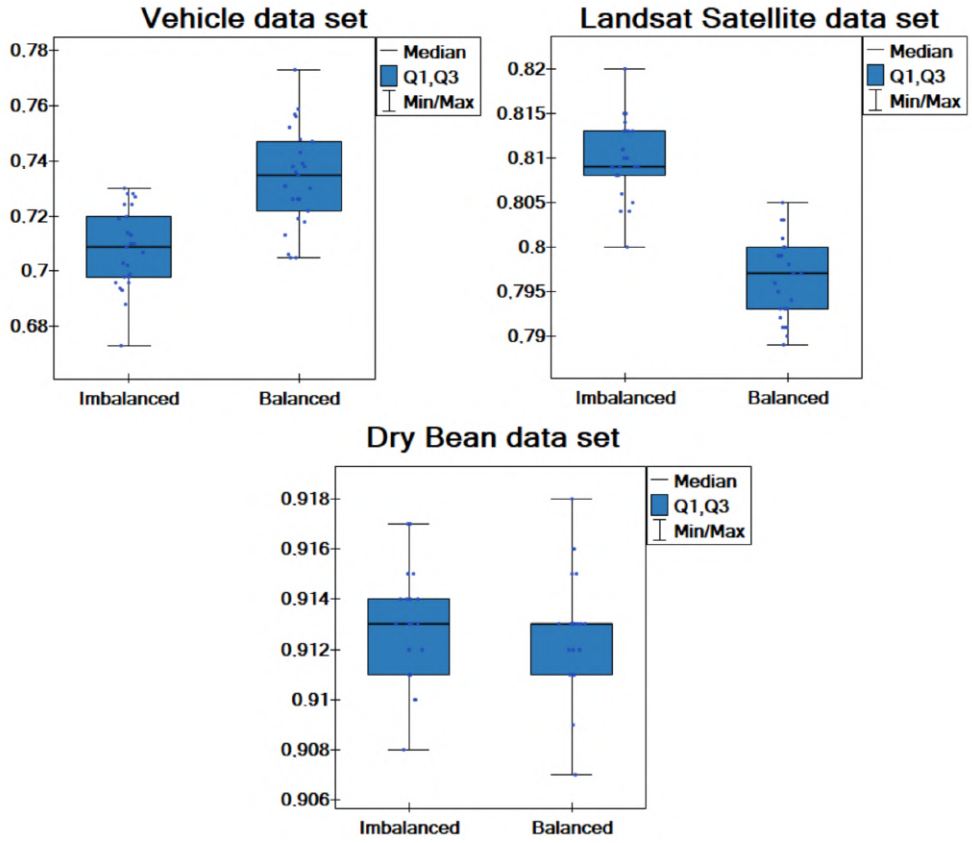


Figure 5. Box-plot chart with (median, the first quartile—Q1, the third quartile—Q3) the value of classification accuracy *acc* for imbalanced and balanced versions of data sets.

3.5. Comparison of Quality of Classification of the Proposed Approach with an Iterative Approach Modeled on Federated Learning

In this section, the results obtained from the approach modeled on federated learning [7,8,11] will be presented. Then a comparison will be made with the results obtained for the proposed approach.

The main difference between the proposed approach and the one based on federated learning is the iterative aggregation of local models. In the proposed approach, local models aggregation occurs only once. The approach modeled on federated learning involves the following steps:

1. Generation of local MLP neural networks based on local tables created analogously as described in Section 2.1. This means that missing attributes are filled in local tables, and artificial objects are generated.
2. The obtained weights and biases from local models are sent to a central server.
3. At the central server, the average of the weights and biases are computed, and the global model obtained is sent back to the local devices.
4. Local devices accept the global model, and once again, trained weights and biases are sent to the central server. Steps 3 and 4 are iterated three times.
5. The global model is post-training on a stratified half of the test set and its accuracy is tested on the remaining half. At another step, the global model is post-training on the other half and tested on the remaining half, after which the classification accuracy is averaged. This is the final step of the process.

As may be noted, an effort was made to provide a fair comparison as both the artificial objects and the post-training process were used in the above approach. An important difference between the proposed approach and the above model is the iterative aggregation of the global model. In addition, the same numbers of artificial objects generated and the same number of neurons in the hidden layer were also analyzed. Of course, the exper-

iments were performed on the same data sets in terms of the degree of dispersion and balanced/imbalanced version. The full results are not given here for the sake of readability and clarity of the paper. Table 5 gives comparison of the results obtained for the proposed approach and the one based on federated learning. In the table, a better result from the two approaches is shown in a bold font. As can be seen, in the overwhelming number of cases, the proposed approach produced better results. Only in thirteen cases for the Vehicle data set did the approach modeled on federated learning produce slightly better results.

Table 5. Comparison of classification accuracy *acc* obtained for the proposed approach (PA) and the approach modeled on federated learning (FL).

Approach		**PA**	**FL**	**PA**	**FL**	**PA**	**FL**	**PA**	**FL**	**PA**	**FL**
Data Set	**No. of Tables**	**No. of Artificially Generated Objects**									
		1	**1**	**2**	**2**	**3**	**3**	**4**	**4**	**5**	**5**
Vehicle imbalanced	3	**0.724**	0.677	**0.688**	0.677	**0.73**	0.673	0.702	**0.709**	0.693	**0.724**
	5	**0.71**	0.681	**0.696**	0.673	**0.728**	0.693	**0.724**	0.717	**0.714**	0.709
	7	0.698	**0.705**	**0.699**	0.693	**0.72**	0.661	**0.719**	0.665	**0.713**	0.701
	9	**0.698**	0.673	**0.707**	0.685	**0.709**	**0.709**	**0.727**	0.697	**0.703**	0.677
	11	**0.694**	0.673	**0.71**	0.673	**0.673**	**0.673**	**0.728**	0.689	**0.696**	0.661
Vehicle balanced	3	**0.73**	0.65	0.705	**0.713**	0.726	**0.752**	**0.735**	0.713	**0.731**	0.689
	5	**0.738**	0.709	**0.748**	0.669	**0.722**	0.701	**0.738**	0.732	**0.726**	0.713
	7	**0.757**	0.709	0.739	**0.748**	**0.756**	0.665	**0.752**	0.736	0.759	**0.764**
	9	**0.743**	0.717	0.718	**0.756**	**0.736**	0.728	0.747	**0.748**	**0.773**	0.748
	11	0.705	**0.709**	0.726	**0.736**	0.706	**0.74**	**0.719**	0.693	0.713	**0.748**
Landsat Satellite imbalanced	3	**0.809**	0.759	**0.809**	0.766	**0.813**	0.773	**0.813**	0.783	**0.808**	0.781
	5	**0.815**	0.766	**0.809**	0.768	**0.82**	0.781	**0.811**	0.78	**0.813**	0.781
	7	**0.814**	0.779	**0.809**	0.77	**0.811**	0.777	**0.81**	0.777	**0.809**	0.769
	9	**0.805**	0.771	**0.813**	0.767	**0.808**	0.784	**0.806**	0.786	**0.809**	0.782
	11	**0.804**	0.771	**0.815**	0.773	**0.81**	0.775	**0.8**	0.781	**0.804**	0.782
Landsat Satellite balanced	3	**0.799**	0.74	**0.799**	0.734	**0.803**	0.743	**0.797**	0.77	**0.803**	0.773
	5	**0.799**	0.74	**0.797**	0.759	**0.805**	0.746	**0.801**	0.777	**0.8**	0.774
	7	**0.8**	0.76	**0.791**	0.752	**0.801**	0.766	**0.793**	0.777	**0.791**	0.773
	9	**0.794**	0.75	**0.793**	0.759	**0.795**	0.762	**0.793**	0.774	**0.796**	0.765
	11	**0.79**	0.757	**0.791**	0.776	**0.789**	0.761	**0.798**	0.763	**0.792**	0.788
Dry Bean imbalanced	3	**0.915**	0.881	**0.917**	0.904	**0.913**	0.883	**0.917**	0.877	**0.914**	0.894
	5	**0.913**	0.872	**0.915**	0.889	**0.912**	0.883	**0.914**	0.893	**0.913**	0.893
	7	**0.911**	0.88	**0.915**	0.899	**0.911**	0.889	**0.91**	0.878	**0.914**	0.873
	9	**0.912**	0.877	**0.913**	0.891	**0.91**	0.875	**0.911**	0.875	**0.913**	0.889
	11	**0.912**	0.887	**0.914**	0.891	**0.908**	0.878	**0.911**	0.889	**0.911**	0.893
Dry Bean balanced	3	**0.915**	0.893	**0.918**	0.91	**0.913**	0.889	**0.916**	0.87	**0.913**	0.89
	5	**0.912**	0.876	**0.916**	0.9	**0.913**	0.884	**0.913**	0.859	**0.913**	0.88
	7	**0.911**	0.878	**0.915**	0.895	**0.913**	0.9	**0.912**	0.884	**0.911**	0.889
	9	**0.913**	0.881	**0.913**	0.89	**0.907**	0.881	**0.911**	0.87	**0.911**	0.881
	11	**0.911**	0.876	**0.911**	0.896	**0.913**	0.872	**0.912**	0.887	**0.909**	0.886

Statistical tests are performed to confirm the significance of the differences in the obtained results *acc* for the proposed approach and the approach modeled on federated learning. The Wilcoxon test using all results from Table 5 is performed. Two dependent groups are analyzed (PA—the proposed approach, FL—the approach modeled on federated learning). The test confirms that differences among the classification accuracy in these two groups are significant ($p = 0.0001$). Additionally, comparative box-plot charts for the values of the classification accuracy are created (Figure 6). The graphs confirm earlier conclusions, and hence it can be said that the proposed method generates better results than the approach modeled on federated learning.

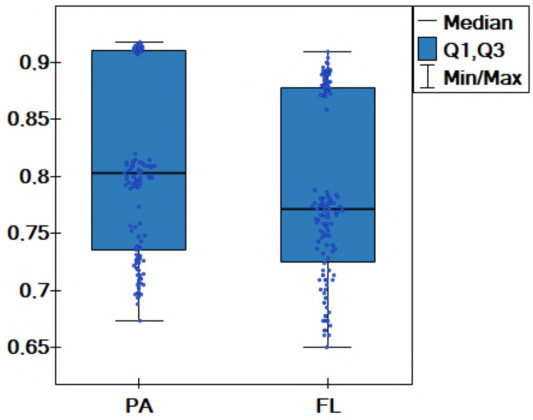


Figure 6. Box-plot chart with (median, the first quartile—Q1, the third quartile—Q3) the value of classification accuracy *acc* for the proposed approach and the approach modeled on federated learning.

4. Conclusions

The paper presented a new method for generating a global MLP model based on dispersed data with different sets of conditional attributes present in local tables. The novelty proposed is the method of generating artificial objects to train local networks with identical structure. An exhaustive comparison of the proposed method has been carried out in terms of the number of artificially generated objects, network structure, data balancing, and degree of data sparseness. The main conclusions are as follows. The greater the number of objects in local tables, the smaller the number of artificially generated objects is sufficient to generate optimal results. For smaller data sets, a greater number of artificial objects (three or four) produces better results. For large data sets, data balancing and the degree of dispersion have no significant impact on the quality of classification. In most cases, a higher number of neurons in the hidden layer gives better results; however, this is very data-dependent and specific. The best results are obtained for the number of neurons in the hidden layer equal to three to five times the number of neurons in the input layer. The paper also confirmed that the proposed method gives better results than the method modeled on federated learning.

In the proposed approach, many aspects should be considered in the future. Among the main plans are to test other ways of aggregating local models and proposing a new method for generating a global training set used in the post-training phase.

Author Contributions: Conceptualization, K.F.M., M.P.-K.; methodology, K.F.M., M.P.-K.; software, K.F.M.; validation, K.F.M., M.P.-K.; formal analysis, M.P.-K., K.F.M.; investigation, M.P.-K., K.F.M.; writing—original draft preparation, M.P.-K.; writing—review and editing, M.P.-K., K.F.M.; visualization, M.P.-K., K.F.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Publicly available data sets were analyzed in this study. This data can be found here: [19].

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Tables A1–A6 show the classification accuracy obtained for different versions of dispersion, different numbers of artificially generated objects, and different numbers of neurons in the hidden layer for Vehicle imbalanced, Vehicle balanced, Landsat Satellite imbalanced, Landsat Satellite balanced, Dry Bean imbalanced, and Dry Bean balanced data sets, respectively.

Table A1. Results of classification accuracy *acc* for the proposed approach with one hidden layer and different number of artificially generated objects—Vehicle imbalanced data set. Designation I is used for the number of neurons in the input layer.

No. of Artificial	No. of	No. of Neurons in Hidden Layer														
Objects	Tables	**0.25 × I**	**0.5 × I**	**0.75 × I**	**1 × I**	**1.5 × I**	**1.75 × I**	**2 × I**	**2.5 × I**	**2.75 × I**	**3 × I**	**3.5 × I**	**3.75 × I**	**4 × I**	**4.5 × I**	**4.75 × I**	**5 × I**
	3	0.283	0.331	0.522	0.664	0.703	0.694	0.717	0.713	0.698	0.663	0.703	**0.724**	0.680	0.711	0.719	0.697
	5	0.281	0.664	0.678	0.690	0.693	0.659	0.699	0.694	0.702	0.664	0.703	0.699	0.694	0.706	**0.710**	0.694
1	7	0.446	0.640	0.664	0.686	0.636	0.659	0.669	0.659	**0.698**	0.682	0.676	0.669	0.673	0.685	0.696	0.685
	9	0.466	0.605	0.572	0.626	0.667	0.660	0.659	**0.698**	0.690	0.654	0.682	0.677	0.675	0.668	0.642	0.692
	11	0.283	0.516	0.559	0.537	0.661	0.652	0.644	0.676	0.636	**0.694**	0.651	0.680	0.648	0.669	0.661	0.664
	3	0.283	0.308	0.664	0.675	0.673	0.680	0.668	0.675	0.685	0.673	0.676	0.685	0.664	0.673	0.685	**0.688**
	5	0.281	0.398	0.644	0.671	0.677	0.665	0.673	0.665	0.690	0.664	0.678	0.682	0.688	**0.696**	0.680	0.690
2	7	0.283	0.672	0.650	0.669	0.696	0.665	0.673	0.686	0.689	0.668	**0.699**	0.688	0.693	0.688	0.673	0.692
	9	0.283	0.667	0.639	0.652	0.660	0.667	0.680	0.690	0.663	**0.707**	0.701	0.673	0.693	0.678	0.678	0.659
	11	0.283	0.283	0.639	0.654	0.699	0.677	0.699	0.676	0.680	0.707	0.692	0.680	0.665	0.696	**0.710**	0.678
	3	0.283	0.675	0.635	0.648	0.647	0.706	0.720	0.698	0.707	0.703	**0.730**	0.723	0.705	0.697	0.694	0.703
	5	0.283	0.630	0.677	0.711	0.680	0.697	0.717	0.714	0.697	0.690	0.714	0.690	**0.728**	0.710	0.710	0.707
3	7	0.283	0.280	0.654	0.684	0.685	0.677	0.688	0.701	**0.720**	0.692	0.709	0.713	0.693	0.692	0.719	0.715
	9	0.283	0.280	0.685	0.682	0.692	0.699	0.673	0.705	**0.709**	0.697	0.701	0.688	0.688	0.682	0.685	0.707
	11	0.283	0.283	0.656	0.668	0.630	0.655	0.668	0.643	0.672	0.659	**0.673**	0.672	0.654	0.671	0.660	0.671
	3	0.283	0.546	0.65	0.638	0.671	0.694	0.681	0.668	0.688	0.66	0.68	0.678	0.667	0.689	0.69	**0.702**
	5	0.283	0.518	0.634	0.657	0.676	0.681	0.664	0.69	0.723	0.696	0.692	0.697	0.69	**0.724**	0.693	0.665
4	7	0.283	0.375	0.684	0.667	0.694	0.68	0.696	0.693	0.668	0.678	0.707	0.678	0.696	**0.719**	0.709	0.718
	9	0.283	0.525	0.673	0.69	0.69	0.706	0.693	0.693	0.718	0.702	0.688	**0.727**	0.689	0.697	0.709	0.685
	11	0.283	0.283	0.661	0.701	**0.728**	0.684	0.707	0.692	0.701	0.699	0.69	0.706	0.705	0.717	0.724	0.698
	3	0.352	0.617	0.63	0.646	0.669	0.671	0.664	0.65	0.686	0.669	0.652	0.685	0.671	**0.693**	0.663	0.635
	5	0.283	0.644	0.692	0.652	0.681	0.701	0.685	0.693	0.673	0.694	0.692	0.702	0.701	0.706	**0.714**	0.71
5	7	0.283	0.391	0.678	0.663	0.692	**0.713**	0.678	0.673	0.699	0.713	0.69	0.69	0.703	0.676	0.706	0.681
	9	0.283	0.52	0.644	0.634	0.685	0.694	0.652	**0.703**	0.698	0.696	0.692	0.686	0.669	0.692	0.681	0.682
	11	0.283	0.283	0.659	0.661	0.657	0.667	0.689	0.667	0.678	0.685	0.661	0.663	**0.696**	0.68	0.673	0.68

Table A2. Results of classification accuracy *acc* for the proposed approach with one hidden layer and different number of artificially generated objects—Vehicle balanced data set. Designation I is used for the number of neurons in the input layer.

No. of Artificial	No. of	No. of Neurons in Hidden Layer														
Objects	Tables	0.25 × I	0.5 × I	0.75 × I	1 × I	1.5 × I	1.75 × I	2 × I	2.5 × I	2.75 × I	3 × I	3.5 × I	3.75 × I	4 × I	4.5 × I	4.75 × I	5 × I
	3	0.283	0.685	0.705	0.692	0.689	0.702	0.720	0.699	0.713	0.705	0.710	0.718	0.726	**0.730**	0.714	0.727
	5	0.374	0.656	0.542	0.680	0.697	0.696	0.727	0.694	0.702	0.693	0.713	0.711	**0.738**	0.694	0.705	0.728
1	7	0.283	0.684	0.715	0.730	0.745	0.751	0.739	0.743	0.731	0.745	0.755	0.738	0.744	0.739	0.739	**0.757**
	9	0.283	0.707	0.685	0.552	0.710	0.707	0.735	0.731	**0.743**	0.738	0.726	0.715	0.732	0.732	0.730	0.709
	11	0.446	0.538	0.654	0.676	0.682	0.660	0.678	0.682	0.656	0.681	0.688	**0.705**	0.677	0.697	0.681	0.696
	3	0.283	0.583	0.651	0.644	0.685	0.696	0.664	0.69	0.675	0.69	0.684	0.672	0.693	0.696	**0.705**	0.69
	5	0.283	0.283	0.686	0.711	0.715	0.711	0.732	0.734	0.724	0.724	0.73	0.722	0.736	0.738	0.731	**0.748**
2	7	0.283	0.567	0.715	0.702	0.713	0.728	0.726	0.71	0.699	0.718	**0.739**	0.738	0.713	0.705	0.703	0.717
	9	0.377	0.689	0.642	0.686	0.682	0.717	0.676	0.715	0.707	0.698	0.689	0.705	0.711	**0.718**	0.685	0.702
	11	0.283	0.697	0.64	0.685	0.671	0.696	0.698	0.711	0.689	0.694	0.709	0.718	0.705	0.688	**0.726**	0.709
	3	0.302	0.685	0.681	0.647	0.697	0.707	0.706	0.710	0.693	0.696	0.690	0.694	0.675	0.689	**0.726**	0.715
	5	0.283	0.677	0.701	0.686	0.675	0.693	0.711	0.713	0.703	0.710	0.703	0.714	0.701	**0.722**	0.718	0.706
3	7	0.283	0.490	0.740	0.715	0.739	0.739	0.734	0.736	0.747	0.735	0.755	0.740	**0.756**	0.740	0.751	0.736
	9	0.283	0.465	0.652	0.702	0.684	0.697	0.709	0.727	0.693	0.701	0.720	0.723	0.699	**0.736**	0.728	0.717
	11	0.283	0.490	0.681	0.657	0.676	0.693	**0.706**	0.698	0.689	0.684	0.698	0.688	0.701	0.696	0.686	0.692
	3	0.283	0.676	0.61	0.71	0.717	0.702	0.728	0.734	**0.735**	0.718	0.726	0.705	0.732	0.731	0.718	0.701
	5	0.283	0.647	0.677	0.696	0.694	0.694	0.703	0.667	0.717	0.717	0.722	0.73	**0.738**	0.736	0.718	0.728
4	7	0.283	0.701	0.554	0.726	0.71	0.73	0.718	0.74	0.727	0.717	0.745	0.73	0.73	0.73	0.748	**0.752**
	9	0.283	0.283	0.701	0.697	0.711	0.711	0.722	**0.747**	0.73	0.744	0.741	0.72	0.738	0.735	0.722	0.727
	11	0.283	0.283	0.669	0.705	0.675	0.688	0.702	0.703	0.697	0.71	**0.719**	0.69	0.717	0.698	0.694	0.713
	3	0.283	0.402	0.699	0.652	0.689	0.714	**0.731**	0.718	0.726	0.709	0.69	0.672	0.673	0.707	0.73	0.697
	5	0.283	0.549	0.655	0.688	0.696	0.685	0.678	0.718	0.681	0.718	0.711	**0.726**	0.722	0.715	0.701	0.71
5	7	0.283	0.486	0.697	0.706	0.694	0.715	**0.759**	0.722	0.741	0.722	0.736	0.747	0.743	0.727	0.732	0.731
	9	0.283	0.323	0.688	0.736	0.718	0.761	0.741	**0.773**	0.739	0.738	0.748	0.705	0.753	0.74	0.741	0.744
	11	0.283	0.283	0.68	0.673	0.682	0.682	0.688	0.678	0.675	0.702	0.697	0.701	0.696	**0.713**	0.706	0.686

Table A3. Results of classification accuracy *acc* for the proposed approach with one hidden layer and different number of artificially generated objects—Landsat Satellite imbalanced data set. Designation I is used for the number of neurons in the input layer.

No. of Artificial	No. of	No. of Neurons in Hidden Layer														
Objects	Tables	**0.25 × I**	**0.5 × I**	**0.75 × I**	**1 × I**	**1.5 × I**	**1.75 × I**	**2 × I**	**2.5 × I**	**2.75 × I**	**3 × I**	**3.5 × I**	**3.75 × I**	**4 × I**	**4.5 × I**	**4.75 × I**	**5 × I**
1	3	0.588	0.789	0.789	0.793	0.806	0.802	0.808	0.805	0.799	0.809	0.805	0.806	**0.809**	0.806	0.802	0.801
	5	0.235	0.793	0.795	0.809	0.804	0.807	0.803	0.798	0.809	0.803	0.810	0.805	0.811	0.813	**0.815**	0.810
	7	0.747	0.795	0.788	0.795	0.796	0.805	**0.814**	0.813	0.808	0.810	0.800	0.808	0.807	0.797	0.799	0.805
	9	0.778	0.791	0.794	0.792	0.791	0.793	0.795	0.801	0.797	0.798	0.800	0.804	0.800	**0.805**	0.801	0.803
	11	0.793	0.787	0.789	0.801	0.794	0.798	0.789	0.799	0.791	**0.804**	0.799	0.802	0.802	0.796	0.802	0.803
2	3	0.765	0.787	0.791	0.803	0.8	0.797	0.796	0.808	0.792	0.804	0.798	**0.809**	0.803	**0.809**	0.803	0.803
	5	0.558	0.489	0.802	0.797	0.803	0.802	0.807	0.803	**0.809**	0.807	0.802	0.797	0.799	0.807	**0.809**	0.802
	7	0.235	0.795	0.798	0.804	0.795	0.797	0.805	0.795	0.799	0.792	**0.809**	0.803	0.807	0.801	0.805	0.803
	9	0.235	0.79	0.794	0.797	0.799	0.797	0.798	0.805	0.8	0.797	0.796	**0.813**	0.801	0.796	0.796	0.803
	11	0.235	0.795	0.792	0.784	0.804	0.802	0.8	0.806	0.805	0.808	0.803	**0.815**	0.81	0.807	**0.815**	0.801
3	3	0.713	0.786	0.798	0.800	0.807	0.796	0.812	0.801	0.804	0.808	0.807	0.809	**0.813**	0.800	0.810	0.806
	5	0.750	0.794	0.796	0.804	0.803	0.805	0.810	0.807	0.811	0.812	0.807	**0.820**	0.810	0.809	0.803	0.804
	7	0.556	0.794	0.799	0.789	0.805	0.798	0.797	0.799	0.790	0.801	0.806	**0.811**	0.799	0.803	0.798	0.801
	9	0.556	0.797	0.801	0.797	0.802	0.800	0.805	0.801	0.801	**0.808**	0.804	0.807	0.802	0.807	0.804	0.805
	11	0.235	0.753	0.794	0.791	0.795	0.803	0.805	0.795	0.796	0.799	**0.810**	0.799	0.804	0.794	0.793	0.792
4	3	0.623	0.796	0.794	0.807	0.809	0.805	0.795	0.801	0.807	0.802	0.811	0.812	0.797	**0.813**	0.809	0.81
	5	0.781	0.798	0.795	0.804	0.79	0.808	0.801	0.805	0.8	0.798	0.81	0.807	0.808	0.808	0.804	**0.811**
	7	0.558	0.78	0.796	0.8	0.791	0.797	0.805	0.795	0.801	**0.81**	0.803	0.8	0.804	0.803	0.796	0.801
	9	0.784	0.797	0.797	0.796	0.794	0.79	0.797	0.783	**0.806**	0.803	0.804	0.796	0.803	0.803	0.796	0.798
	11	0.235	0.792	0.793	0.794	0.795	0.794	0.797	0.796	0.793	0.799	0.795	0.784	0.794	0.796	0.794	**0.8**
5	3	0.235	0.783	0.786	0.794	0.79	0.803	0.791	0.792	0.803	0.805	0.789	0.802	0.797	0.807	0.803	**0.808**
	5	0.561	0.794	0.8	0.793	0.811	0.809	0.806	0.791	0.809	0.806	0.812	0.812	0.801	**0.813**	**0.813**	0.809
	7	0.66	0.79	0.796	0.799	0.799	**0.809**	0.803	0.797	0.805	0.806	0.8	0.801	0.802	0.808	0.806	0.803
	9	0.559	0.792	0.802	0.792	0.799	0.792	0.805	**0.809**	0.8	0.801	0.803	0.806	0.801	0.799	0.802	0.801
	11	0.235	0.789	0.799	0.798	0.79	0.797	0.799	0.802	0.8	0.801	0.801	0.803	0.801	0.8	**0.804**	0.794

Table A4. Results of classification accuracy *acc* for the proposed approach with one hidden layer and different number of artificially generated objects—Landsat Satellite balanced data set. Designation I is used for the number of neurons in the input layer.

No. of Artificial	No. of	No. of Neurons in Hidden Layer														
Objects	Tables	**0.25 × I**	**0.5 × I**	**0.75 × I**	**1 × I**	**1.5 × I**	**1.75 × I**	**2 × I**	**2.5 × I**	**2.75 × I**	**3 × I**	**3.5 × I**	**3.75 × I**	**4 × I**	**4.5 × I**	**4.75 × I**	**5 × I**
	3	0.767	0.794	0.791	0.782	0.787	0.792	0.787	0.788	0.779	0.787	**0.799**	0.796	0.786	0.789	0.780	0.786
	5	0.238	0.740	0.778	0.781	0.785	0.781	0.793	0.788	0.793	0.795	0.789	0.796	0.787	0.785	0.790	**0.799**
1	7	0.417	0.776	0.789	0.793	0.795	0.800	0.793	0.784	0.793	0.794	0.798	0.791	0.789	0.783	**0.800**	0.796
	9	0.238	0.782	0.787	0.786	0.790	0.780	0.785	0.782	0.781	0.786	0.780	0.785	0.782	0.787	0.787	**0.794**
	11	0.238	0.741	0.780	0.781	0.785	0.788	0.787	0.789	0.788	0.789	0.781	0.782	**0.790**	0.783	0.784	0.789
	3	0.238	0.775	0.779	0.789	0.782	0.781	0.791	0.79	0.791	0.789	0.788	0.789	0.788	0.785	**0.799**	0.791
	5	0.238	0.787	0.793	0.795	0.778	0.786	0.793	0.789	0.794	0.796	0.791	0.795	0.788	**0.797**	0.791	0.787
2	7	0.238	0.788	0.786	0.787	0.785	0.78	0.781	0.781	0.778	0.79	**0.791**	0.767	0.782	0.782	0.777	0.785
	9	0.689	0.789	0.783	0.791	0.787	**0.793**	0.777	0.786	0.785	0.789	0.792	0.792	0.787	0.787	0.789	0.781
	11	0.563	0.772	0.783	0.785	0.78	0.779	0.782	0.789	0.782	0.783	0.772	0.785	**0.791**	0.78	0.785	0.789
	3	0.562	0.782	0.786	0.781	0.778	0.779	0.787	0.794	0.780	0.781	0.791	0.773	0.801	0.787	**0.803**	0.790
	5	0.238	0.785	0.787	0.795	0.791	0.788	0.788	0.790	0.798	0.786	0.784	0.784	**0.805**	0.782	0.787	0.790
3	7	0.238	0.792	0.786	0.790	0.785	0.793	0.797	0.777	0.786	0.788	0.792	**0.801**	0.798	0.789	0.780	0.798
	9	0.564	0.781	0.784	0.786	**0.795**	0.791	0.791	0.787	0.794	0.794	0.787	**0.795**	0.792	0.787	0.794	0.792
	11	0.567	0.775	0.780	0.782	0.788	0.780	0.783	0.786	0.785	0.780	0.779	0.786	0.788	0.784	**0.789**	0.773
	3	0.753	0.774	0.778	0.788	0.788	0.783	0.779	0.788	0.788	0.781	0.783	0.787	0.775	0.787	0.796	**0.797**
	5	0.568	0.787	0.788	0.79	0.794	0.786	0.786	0.793	0.79	0.799	0.79	0.781	0.79	0.797	0.797	**0.801**
4	7	0.566	0.773	0.78	0.787	0.792	0.784	0.771	0.79	0.781	0.783	0.791	0.775	**0.793**	0.79	0.779	0.775
	9	0.238	0.773	0.788	0.778	0.775	0.78	0.787	0.78	0.78	0.763	**0.793**	0.771	0.785	0.791	0.787	0.784
	11	0.566	0.785	0.787	0.78	0.784	0.781	0.782	0.782	0.791	0.778	0.784	0.776	0.79	0.776	0.776	**0.798**
	3	0.569	0.781	0.794	0.793	0.787	0.779	0.794	0.799	0.797	**0.803**	0.8	0.788	0.794	0.797	0.792	0.798
	5	0.676	0.777	0.781	0.784	0.785	**0.8**	0.792	0.795	0.798	0.784	0.786	0.781	0.792	0.794	0.793	0.792
5	7	0.734	0.779	0.781	0.78	0.781	0.778	0.782	0.778	0.787	0.775	0.78	0.779	**0.791**	0.789	0.788	0.769
	9	0.401	0.788	0.78	0.787	0.792	0.79	0.794	0.787	0.784	0.793	0.789	**0.796**	0.772	0.793	0.783	0.78
	11	0.238	0.759	0.785	0.785	0.787	0.784	0.79	**0.792**	0.788	0.791	0.786	0.786	0.787	0.786	0.782	0.788

Table A5. Results of classification accuracy *acc* for the proposed approach with one hidden layer and different number of artificially generated objects —Dry Bean imbalanced data set. Designation I is used for the number of neurons in the input layer.

No. of Artificial	No. of	No. of Neurons in Hidden Layer														
Objects	Tables	**0.25 × I**	**0.5 × I**	**0.75 × I**	**1 × I**	**1.5 × I**	**1.75 × I**	**2 × I**	**2.5 × I**	**2.75 × I**	**3 × I**	**3.5 × I**	**3.75 × I**	**4 × I**	**4.5 × I**	**4.75 × I**	**5 × I**
	3	0.877	0.891	0.892	0.902	0.902	0.904	0.904	0.910	0.912	0.910	0.910	0.912	0.912	0.913	**0.915**	**0.915**
	5	0.541	0.888	0.894	0.900	0.900	0.907	0.903	0.907	0.907	0.909	**0.913**	0.911	0.911	0.912	0.911	0.911
1	7	0.885	0.887	0.893	0.894	0.901	0.900	0.902	0.903	0.905	0.906	0.909	0.910	**0.911**	**0.911**	**0.911**	**0.911**
	9	0.879	0.890	0.888	0.893	0.897	0.899	0.903	0.904	0.905	0.907	0.906	0.907	0.908	0.910	**0.912**	0.911
	11	0.882	0.890	0.892	0.893	0.899	0.899	0.902	0.904	0.905	0.908	0.907	0.908	0.910	**0.912**	**0.912**	0.910
	3	0.886	0.889	0.903	0.905	0.911	0.914	0.911	0.912	0.912	0.912	0.916	0.915	0.916	0.916	**0.917**	**0.917**
	5	0.822	0.89	0.904	0.899	0.902	0.905	0.905	0.91	0.911	0.911	0.911	0.912	0.914	0.914	0.914	**0.915**
2	7	0.88	0.895	0.903	0.896	0.904	0.908	0.908	0.907	0.908	0.911	0.911	0.911	0.911	0.914	**0.915**	0.913
	9	0.884	0.888	0.896	0.894	0.899	0.902	0.905	0.905	0.905	0.906	0.91	0.912	0.912	0.912	0.911	**0.913**
	11	0.884	0.891	0.897	0.894	0.897	0.899	0.902	0.904	0.907	0.906	0.908	0.912	0.909	0.913	0.911	**0.914**
	3	0.885	0.895	0.897	0.898	0.904	0.905	0.907	0.908	0.910	0.911	0.911	**0.913**	0.912	0.912	0.912	**0.913**
	5	0.886	0.888	0.890	0.902	0.902	0.906	0.905	0.906	0.907	0.909	0.909	0.910	0.910	0.910	**0.912**	**0.912**
3	7	0.804	0.887	0.896	0.892	0.902	0.903	0.902	0.908	0.907	0.906	0.909	0.907	0.909	**0.911**	**0.911**	**0.911**
	9	0.520	0.886	0.890	0.895	0.896	0.899	0.897	0.905	0.904	0.902	0.907	0.907	0.907	**0.910**	0.909	0.909
	11	0.633	0.886	0.891	0.896	0.897	0.897	0.900	0.900	0.902	0.902	0.903	0.905	0.905	0.906	0.907	**0.908**
	3	0.79	0.894	0.895	0.9	0.907	0.907	0.909	0.91	0.909	0.91	0.913	0.913	0.913	0.914	0.914	**0.917**
	5	0.884	0.893	0.899	0.9	0.903	0.903	0.903	0.907	0.909	0.911	0.909	0.911	0.911	0.91	0.911	**0.914**
4	7	0.866	0.887	0.889	0.896	0.901	0.898	0.898	0.905	0.906	0.905	0.906	0.909	0.909	0.908	**0.91**	**0.91**
	9	0.887	0.886	0.892	0.892	0.898	0.902	0.903	0.903	0.905	0.906	0.91	0.908	0.91	0.909	**0.911**	**0.911**
	11	0.78	0.889	0.895	0.896	0.898	0.899	0.9	0.901	0.903	0.906	0.909	0.908	0.908	0.909	0.909	**0.911**
	3	0.876	0.892	0.901	0.898	0.901	0.91	0.908	0.911	0.913	0.909	0.912	**0.914**	**0.914**	**0.914**	**0.914**	0.911
	5	0.882	0.887	0.893	0.897	0.903	0.903	0.906	0.909	0.907	0.908	0.91	0.91	0.911	0.912	**0.913**	**0.913**
5	7	0.883	0.89	0.893	0.895	0.898	0.902	0.901	0.904	0.906	0.908	0.91	0.911	0.91	0.913	0.911	**0.914**
	9	0.883	0.887	0.889	0.894	0.899	0.899	0.901	0.903	0.906	0.904	0.907	0.908	0.911	**0.913**	0.91	0.911
	11	0.856	0.889	0.892	0.892	0.897	0.898	0.897	0.907	0.903	0.902	0.906	0.908	0.908	0.91	**0.911**	0.908

Table A6. Results of classification accuracy *acc* for the proposed approach with one hidden layer and different number of artificially generated objects—Dry Bean balanced data set. Designation I is used for the number of neurons in the input layer.

No. of Artificial	No. of	No. of Neurons in Hidden Layer														
Objects	Tables	**0.25 × I**	**0.5 × I**	**0.75 × I**	**1 × I**	**1.5 × I**	**1.75 × I**	**2 × I**	**2.5 × I**	**2.75 × I**	**3 × I**	**3.5 × I**	**3.75 × I**	**4 × I**	**4.5 × I**	**4.75 × I**	**5 × I**
	3	0.421	0.891	0.897	0.898	0.901	0.905	0.904	0.910	0.910	0.911	0.911	0.912	0.911	0.914	0.913	**0.915**
	5	0.883	0.891	0.894	0.900	0.902	0.904	0.904	0.909	0.909	0.909	0.910	0.910	0.909	0.910	0.911	**0.912**
1	7	0.657	0.888	0.893	0.894	0.897	0.897	0.900	0.904	0.905	0.905	0.907	0.908	0.909	0.909	**0.911**	**0.911**
	9	0.881	0.885	0.894	0.896	0.898	0.898	0.901	0.903	0.905	0.907	0.908	0.910	0.907	0.911	**0.913**	0.912
	11	0.871	0.887	0.892	0.894	0.895	0.903	0.903	0.902	0.903	0.908	0.909	0.909	0.909	0.910	**0.911**	0.909
	3	0.897	0.902	0.905	0.906	0.912	0.91	0.914	0.916	0.914	0.916	0.916	0.917	**0.918**	0.916	0.917	**0.918**
	5	0.886	0.901	0.906	0.905	0.907	0.909	0.909	0.911	0.907	0.91	0.913	0.915	0.914	0.914	**0.916**	0.913
2	7	0.634	0.892	0.903	0.901	0.907	0.904	0.905	0.909	**0.915**	0.911	0.914	0.912	0.914	0.913	0.91	0.914
	9	0.863	0.894	0.896	0.9	0.903	0.906	0.908	0.908	0.908	0.907	0.912	0.91	0.912	0.912	0.912	**0.913**
	11	0.878	0.895	0.894	0.892	0.902	0.901	0.901	0.909	0.905	0.908	0.906	0.91	0.909	0.91	0.91	**0.911**
	3	0.413	0.890	0.898	0.901	0.904	0.908	0.909	0.908	0.909	0.908	0.910	0.912	0.912	**0.913**	**0.913**	0.911
	5	0.580	0.889	0.891	0.896	0.899	0.906	0.907	0.905	0.906	0.908	0.909	0.909	**0.913**	**0.913**	0.912	0.911
3	7	0.887	0.889	0.894	0.900	0.900	0.903	0.904	0.905	0.906	0.909	0.910	0.912	0.910	**0.913**	0.912	0.912
	9	0.606	0.884	0.892	0.893	0.897	0.903	0.899	0.901	0.901	0.904	0.904	0.905	0.906	0.904	**0.907**	0.906
	11	0.734	0.891	0.892	0.894	0.898	0.898	0.901	0.902	0.905	0.906	0.906	0.906	0.909	0.911	0.910	**0.913**
	3	0.884	0.893	0.894	0.897	0.905	0.904	0.907	0.909	0.911	0.91	0.914	0.911	0.913	0.913	0.914	**0.916**
	5	0.884	0.892	0.894	0.893	0.902	0.904	0.902	0.904	0.909	0.909	0.907	0.909	0.912	0.911	**0.913**	**0.913**
4	7	0.884	0.888	0.894	0.897	0.9	0.9	0.903	0.905	0.906	0.907	0.909	0.909	0.907	0.909	0.91	**0.912**
	9	0.854	0.888	0.886	0.89	0.901	0.902	0.903	0.901	0.904	0.904	0.902	0.905	0.91	0.909	0.91	**0.911**
	11	0.844	0.89	0.893	0.895	0.896	0.905	0.901	0.906	0.905	0.906	0.907	0.909	0.909	**0.912**	0.911	**0.912**
	3	0.883	0.889	0.899	0.901	0.906	0.908	0.907	0.908	0.91	0.909	0.91	0.912	0.912	0.911	**0.913**	**0.913**
	5	0.884	0.895	0.896	0.9	0.905	0.907	0.903	0.906	0.906	0.907	0.911	0.91	0.911	0.912	**0.913**	0.912
5	7	0.801	0.885	0.889	0.896	0.901	0.905	0.9	0.904	0.905	0.906	0.908	0.907	**0.911**	**0.911**	0.91	0.91
	9	0.882	0.888	0.891	0.892	0.898	0.9	0.9	0.903	0.903	0.901	0.907	0.908	0.907	**0.911**	0.91	**0.911**
	11	0.425	0.891	0.892	0.892	0.897	0.897	0.897	0.905	0.904	0.905	0.907	**0.909**	0.907	0.907	**0.909**	**0.909**

References

1. Bazan, J.G.; Drygaś, P.; Zaręba, L.; Molenda, P. A new method of building a more effective ensemble classifiers. In Proceedings of the 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Glasgow, UK, 19–24 July 2020; pp. 1–6.
2. Piwowarczyk, M.; Muke, P.Z.; Telec, Z.; Tworek, M.; Trawiński, B. Comparative analysis of ensembles created using diversity measures of regressors. In 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada, 11–14 October 2020; pp. 2207–2214.
3. Muzammal, M.; Talat, R.; Sodhro, A.H.; Pirbhulal, S. A multi-sensor data fusion enabled ensemble approach for medical data from body sensor networks. *Inf. Fusion* **2020**, *53*, 155–164. [CrossRef]
4. Przybyła-Kasperek, M. Comparison of Dispersed Decision Systems with Pawlak Model and with Negotiation Stage in Terms of Five Selected Fusion Methods. In Proceedings of the Computational Collective Intelligence ICCCI 2018 10th International Conference, ICCCI 2018, Bristol, UK, 5–7 September 2018; pp. 301–310. [CrossRef]
5. Seydi, S.T.; Saeidi, V.; Kalantar, B.; Ueda, N.; van Genderen, J.L.; Maskouni, F.H.; Aria, F.A. Fusion of the multisource datasets for flood extent mapping based on ensemble convolutional neural network (CNN) model. *J. Sens.* **2022**, *2022*, 2887502. [CrossRef]
6. Firouzi, R.; Rahmani, R.; Kanter, T. Federated learning for distributed reasoning on edge computing. *Procedia Comput. Sci.* **2021**, *184*, 419–427. [CrossRef]
7. Połap, D. Fuzzy consensus with federated learning method in medical systems. *IEEE Access* **2021**, *9*, 150383–150392. [CrossRef]
8. Mothukuri, V.; Parizi, R.M.; Pouriyeh, S.; Huang, Y.; Dehghantanha, A.; Srivastava, G. A survey on security and privacy of federated learning. *Future Gener. Comput. Syst.* **2021**, *115*, 619–640. [CrossRef]
9. Marfo, K.F.; Przybyła-Kasperek, M. Radial basis function network for aggregating predictions of k-nearest neighbors local models generated based on independent data sets. *Procedia Comput. Sci.* **2022**, *207*, 3234–3243. [CrossRef]
10. Przybyła-Kasperek, M.; Marfo, K.F. Neural network used for the fusion of predictions obtained by the K-Nearest neighbors algorithm based on independent data sources. *Entropy* **2021**, *23*, 1568. [CrossRef] [PubMed]
11. Venkatesha, Y.; Kim, Y.; Tassiulas, L.; Panda, P. Federated learning with spiking neural networks. *IEEE Trans. Signal Process.* **2021**, *69*, 6183–6194. [CrossRef]
12. Senousy, Z.; Abdelsamea, M.M.; Mohamed, M.M.; Gaber, M.M. 3E-Net: Entropy-based elastic ensemble of deep convolutional neural networks for grading of invasive breast carcinoma histopathological microscopic images. *Entropy* **2021**, *23*, 620. [CrossRef] [PubMed]
13. Glorot, X.; Bordes, A.; Bengio, Y. Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Ft. Lauderdale, FL, USA, 11–13 April 2011; pp. 315–323.
14. Li, X.; Li, X.; Pan, D.; Zhu, D. On the learning property of logistic and softmax losses for deep neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 4739–4746.
15. Kingma, D.P.; Ba, J. In Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations, ICLR, San Diego, CA, USA, 7–9 May 2015.
16. Mannor, S.; Peleg, D.; Rubinstein, R. The cross entropy method for classification. In Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 7–11 August 2005; pp. 561–568.
17. Schapire, R.E. *Explaining Adaboost. Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 37–52.
18. Siebert, J.P. *Vehicle Recognition Using Rule Based Methods*; Turing Institute: London, UK, 1987.
19. Asuncion, A.; Newman, D.J. *UCI Machine Learning Repository*; University of Massachusetts Amherst: Amherst, MA, USA, 2007. Available online: https://archive.ics.uci.edu (accessed on 10 March 2023).
20. Koklu, M.; Ozkan, I.A. Multiclass classification of dry beans using computer vision and machine learning techniques. *Comput. Electron. Agric.* **2020**, *174*, 105507. [CrossRef]
21. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
22. Ingrid, R.; Zdravko, M. An introduction to the weka data mining system. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, Seattle, WA, USA, 8–11 March 2017; p. 742.

Publication [P6]

Marfo K.F.,Przybyła-Kasperek M., Sulikowski P. Fragmented Image Classification Using Local and Global Neural Networks: Investigating the Impact of the Quantity of Artificial Objects on Model Performance *International Conference on Computational Science*, 14838:280-194, 2024.
URL: https://link.springer.com/chapter/10.1007/978-3-031-63783-4_21.
DOI: 10.1007/978-3-031-63783-4_21.
$\text{MEiN}_{2024} = 140$
Number of citations:

- according to Web of Science: 0
- according to Google Scholar: 0

Contributor	Description of main tasks
Kwabena Frimpong Marfo	- investigation - conceptualization and methodology - result analysis - implementation of proposed algorithm - manuscript: review and editing - visualization: creating all figures in manuscript
Małgorzata Przybyła-Kasperek	- conceptualization and methodology - result analysis - manuscript: original draft preparation - manuscript: review and editing - visualization: creating all figures in manuscript
Piotr Sulikowski	- result analysis - manuscript: review and editing

Fragmented Image Classification Using Local and Global Neural Networks: Investigating the Impact of the Quantity of Artificial Objects on Model Performance

Kwabena Frimpong Marfo[1][0000−0003−2226−9097], Małgorzata Przybyła-Kasperek[1][0000−0003−0616−9694], and Piotr Sulikowski[2][0000−0002−8704−8925]

[1] University of Silesia in Katowice, Institute of Computer Science, Będzińska 39, 41-200 Sosnowiec, Poland
{kmarfo,malgorzata.przybyla-kasperek}@us.edu.pl
[2] West Pomeranian University of Technology in Szczecin, Faculty of Computer Science and Information Technology, ul. Żołnierska 49, 71-210 Szczecin, Poland
piotr.sulikowski@zut.edu.pl

Abstract. This paper addresses the challenge of classifying objects based on fragmented data, particularly when dealing with characteristics extracted from images captured from various angles. The complexity increases when dealing with fragmented images that may partially overlap. The paper introduces a classification model utilizing neural networks, specifically multilayer perceptron (MLP) networks. The key concept involves generating local models based on local tables comprising characteristics extracted from fragmented images. Since the local tables may have different sets of attributes due to varying perspectives, missing attributes in the tables are imputed by introducing artificial objects. The local models, now with identical structures are created and the aggregation of these models into a global model is carried out using weighted averages. The model's efficacy is evaluated against existing literature methods using various metrics, demonstrating superior performance in terms of F-measure and balanced accuracy. Notably, the paper investigates the impact of the number of generated artificial objects on classification quality, revealing that a higher number generally improves results.

Keywords: Fragmented Image Classification · Neural Networks · Artificial Objects · Characteristics Generated From Images.

1 Introduction

Every so often in computer vision and object recognition tasks, the goal is not necessarily to create a virtual representation of an object, but to assign it to a certain class, based on its characteristics. Such a situation occurs in instances such as recognizing the architectural style of a building, the type of vehicle or type of land based on a satellite image. The situation becomes more complicated

when not a single image represents an entire object, but rather many fragmented images that may partially overlap. Here, one can think about a set of cameras that perceive an image of an object from different angles. Fragmented images can partially overlap – the cameras can observe (in some part) the same fragment of the object. In this paper, an assumption of not having fragmented images as such, but rather characteristics that have been extracted from these images is made. These object characteristics are stored in decision tables, also known as local tables and may contain common attributes and objects. By common objects in tables, we mean a situation where characteristics extracted from images of the same object are stored in different tables. There may be inconsistencies among tables in that an image in question may have been distorted in some way, resulting in a completely different value on conditional attributes or even decision attribute for the same object.

The paper proposes a classification model based on such fragmented data. The main idea of the model is to use neural networks (specifically MLP networks) to generate local models based on each local table. These models are then aggregated into a global model using trained weights from the local models. However, aggregation of MLP networks is not possible to realize without the local models having the same structure. This constraint can be satisfied by ensuring the presence of the same conditional attributes in all local tables, which, of course is not originally fulfilled due to different cameras observing different parts of an object and, consequently, different set of characteristics being stored in local tables. So, to achieve homogeneity in local tables, it is necessary to modify them before generating local models. This is done by generating artificial objects – supplemented with values for missing attributes. After such modification, local models with identical structures are built from local tables. Local models are then aggregated into a global model. Finally, the global model is refined using a small set of objects.

Figure 1 shows the stages of building the global model. In the first and second steps, characteristics of objects are extracted from fragmented images and local decision tables are created with different sets of conditional attributes – sets of attributes are not necessarily disjoint. It should be noted that this part is not addressed directly in this paper, since the data used retains characteristics of objects extracted from images and were obtained from the repository. Then, in order to unify the local tables, values are imputed for missing attributes in all local tables. For an original object, more than one artificial object can be created, making the cardinality objects in local tables dynamic. The fourth step describes building local neural networks, and the fifth aggregates these local networks into a global model – weights from local models are used for this purpose. Finally, this global model is trained using a small set of objects.

The main contribution of this paper is a proposal of a classification model based on characteristics obtained from fragmented images. Comparing the classification quality of the proposed model with known methods from the literature, it was shown that the proposed model, on average, generates better F-measure and balanced accuracy results. An important result of the paper is to examine

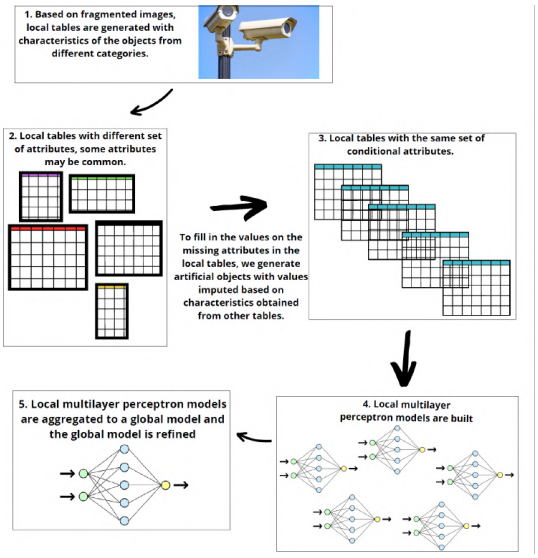


Fig. 1. Global model generation stages.

the effect of the number of artificial objects generated on classification quality. Here, usually a higher number of objects improves the quality of classification. The paper also provides some guidance for which data the proposed model is suitable.

An effective and excellent model for image recognition is convolution neural networks [15]. There are many applications of image-based object recognition, among which are handwriting recognition [13], X-ray analysis [5], plant disease identification [3], facial emotion recognition [1]. However, it should be admitted that there are not many papers dedicated to the subject of processing fragmented images. Papers on fragmented images are usually concerned with completing the object that are presented in a fragmentary way on the image – examples are cracks in a roadway [14] or fragments of a plant leaf [2], coloring and fragmentation of image where the objects are located [9]. Identifying fragmented images is an infrequently addressed area of study. This challenge often involves working with sets of images captured from diverse perspectives, where each image is obtained by a distinct camera viewing the object from different angles [11]. When dealing with fragmented image data, recognizing the depicted object becomes more challenging. In a related paper [7], the proposed methodology involves assembling fragmented portions of photos to reconstruct a complete image, aligning and merging components to form a cohesive whole. It is important to note that this approach assumes non-overlapping fragmented data for effective implementation. The first research on classification based on dispersed and fragmented images was presented in the paper [10]. In this paper we present

an approach that builds a common model which given the characteristics of an object, can recognize the class from which it comes from.

In Section 2, the proposed classification model is described. Section 3 addresses the data sets that were used and presents the conducted experiments and discussion on obtained results. Section 4 is on conclusions.

2 Model and methods

There is an assumption that features are extracted from fragmented images and stored in a tabular form. More formally, some characteristics of images are available in a dispersed form, that is, in the form of a set of local tables. A set of decision tables $D_i = (U_i, A_i, d)$, $i \in \{1, \ldots, n\}$ is available, where U_i is the universe comprising objects – images; A_i is a set of attributes that describe the image; d is a decision attribute – object shown in the image; $A = \bigcup_{i=1}^{n} A_i$ is the union of attributes present in all local tables. Objects and attributes in local tables can be different but some may be common.

The aim is to to generate local neural network models (MLP models) based on each local table. To construct a global model, the structure of such local models must be identical, and this can only be achieved if all local tables have the same sets of attributes. Each table D_i is modified so that the full set of attributes A is included. This is done by generating new objects with completed values on the missing attributes. Suppose the object $\bar{x} \in U_i$ has a decision value v, $d(\bar{x}) = v$, $v \in V^d$, where V^d is the set of values of the decision attribute d and $b \in A \setminus A_i$. For each decision table $D_j, i \neq j$ for which $b \in A_j$ the following values are computed: $MIN^b_{j,v} = min_{x \in U_j, d(x)=v}\ b(x)$, $MAX^b_{j,v} = max_{x \in U_j, d(x)=v}\ b(x)$, $AVG^b_{j,v} = avg_{x \in U_j, d(x)=v}\ b(x)$, $MED^b_{j,v} = median_{x \in U_j, d(x)=v}\ b(x)$. In this way, individual values assigned to attribute b for each local table are derived. The final value, which completes the object $\bar{x}$ in table D_i is determined by applying one of four statistical measures (minimum, maximum, mean, or median) to the local values obtained in the previous step. Consequently, there are sixteen potential combinations with one chosen randomly for determining the value of attribute b. As an illustration, consider a scenario where the maximum is selected for calculating local values, and the median is chosen for the aggregate value. In this case, the determination of the value for attribute b is as follows: $b(\bar{x}) = med_{D_j : b \in A_j} MAX^b_{j,v}$. This method is repeated for each attribute that does not belong to the set A_i but occurs in other local tables. By the procedure described above, one can generate several artificial objects based on an original object. A parameter k is used to determine the number of artificial objects generated based on a single original object. This expanded approach has been tested, and the corresponding results are detailed in the experimental section of the paper. In this way, a set of modified local tables $\bar{D}_i = (\bar{U}_i, A, d)$, $i \in \{1, \ldots, n\}$ with equal sets of attributes is obtained.

The local tables $\bar{D}_i$ are used in subsequent steps for training MLP neural networks. The input layer is defined as the cardinality of A. The number of neurons in the output layer corresponds to the number of decision classes, where

each neuron determines the probability of the test object belonging to a specific decision class. In the experimental section, the consideration is given to one or two hidden layers. The number of neurons in the hidden layer is determined proportionally to the number of neurons in the input layer, exploring different proportions ranging from 0.25 to 5 times the number of input layer neurons. In the case of two hidden layers, all combinations of neuron numbers are explored, with the first layer being chosen from the set $\{0.25\times I, 0.5\times I, 0.75\times I, 1\times I, 1.5\times I, 1.75\times I, 2\times I, 2.5\times I, 2.75\times I, 3\times I, 3.5\times I, 3.75\times I, 4\times I, 4.5\times I, 4.75\times I, 5\times I\}$, and the second layer chosen from the set $\{1\times I, 2\times I, 3\times I, 4\times I, 5\times I\}$ where I is the number of neurons in the input layer. The ReLU (Rectified Linear Unit) function is employed as the activation function for the hidden layer. For the output layer, the softmax activation function is utilized. The neural network is trained using the back-propagation method, specifically employing a gradient descent method with an adaptive step size. The model employs the categorical cross-entropy loss function in conjunction with the Adam optimizer for optimal performance.

Since all the local models created have the same structure, the global model is created by a weighted sum of the trained weights from local models. Prior to this aggregation, weights from each local model are adjusted by the formula: $\omega_i = ln(\frac{1-e_i}{e_i})$, where e_i is the classification error of the $i-$th local model on the training set $\bar{U}_i$. In summary, the global model is created as follows: initially, the network's structure is specified, then its weights assigned based on the weighted sum of weights from the local models. The final stage involves retraining the global network. For this step, training objects must possess values of all attributes A. This can be a certain set of examples/objects that an expert will describe and classify by capturing all the characteristics of an object at once. In the experiments conducted in this paper, such a validation set derived from the test set.

3 Data sets and results

3.1 Data and measures

The proposed system was tested on three data sets from the UC Irvine Machine Learning Repository [6, 8, 12]. Vehicle Silhouettes – aims to classify vehicle silhouettes into one of four types, considering characteristics extracted from images taken from various angles: eighteen quantitative attributes, four decision classes, 846 objects (592 training, 254 test set). Landsat Satellite – involves classifying earth types in satellite images based on multispectral pixel values in a 3×3 neighborhood: thirty-six quantitative attributes, six decision classes, 6435 objects (4435 training, 1000 test set). Dry Bean – focuses on classifying types of beans using characteristics extracted from high-resolution images subjected to segmentation and feature extraction stages: seventeen quantitative attributes, seven decision classes, 13611 objects (9527 training, 4084 test set).

The data pre-processing involved random dispersion into 3, 5, 7, 9, and 11 local tables. Each local table included a reduced set of attributes with all objects

from the original table. The data sets exhibited imbalance, with varying object counts across decision classes in both training and test sets. Two variants were considered for each data set: experiments on dispersed imbalanced data and on balanced data modified using the Synthetic Minority Over-sampling Technique (SMOTE) method [4].

The quality of classification was evaluated based on the test set with the following accuracy measures. Classification accuracy measure (*acc*) – a fraction of the total number of objects in the test set that were classified correctly; Recall – an assessment of the classifier's ability to correctly recognize a given class; Precision (Prec.) – a measure of how often the classifier does not make a mistake when classifying an object to a given class, F-measure (F-m.) – an assessment of the classifier's ability to keeping accuracies balanced. F-measure $= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$; Balanced accuracy – an average value of Recall for all decision classes. Balanced accuracy (*bacc*) ensures that the performance assessment considers the classification accuracy of all classes equally.

The effect of the number of artificial objects created based on the original object from the local table on the quality of classification was tested. The following numbers of artificial objects used were examined $\{1, 2, 3, 4, 5\}$. During the experiments, different structures of local networks with one or two hidden layers were also tested. Moreover, different number of neurons in hidden layers were studied. The following values were tested: for the first hidden layer $\{0.25, 0.5, 0.75, 1, 1.5, 1.75, 2, 2.5, 2.75, 3, 3.5, 3.75, 4, 4.5, 4.75, 5\}\times$ the number of neurons in the input layer; for the second hidden layer $\{1, 2, 3, 4, 5\}\times$ the number of neurons in the input layer. A validation set was obtained by dividing the original test set randomly but in a stratified manner into two equal parts. First, one part is used as the validation set (for re-training process) and the second part is used to assess the quality of classification. Then the roles reverse as the second part acts as the validation set. Finally, both results are averaged. Each experiment is repeated three times; in the following tables, all results given are the average of these three runs.

3.2 Results analysis

Due to space limit, results obtained for all parameters are not shown (however, they will be made available upon request sent to the authors). Tables 1, 2 and 3 show the best (in terms of classification accuracy) results obtained. The tables also show in bold the best result for each of the considered data sets.

The proposed approach was compared with three other approaches. The first approach (MLP ensemble) uses a homogeneous ensemble of MLP networks. The final decision was determined by soft voting since networks generated from the local tables could be not aggregated due to their different structures. The second approach uses an ensemble of classifiers (KNN, DT, NB). This ensemble of classifiers method consists of creating three base classifiers: $k-$nearest neighbors, decision tree and naive bayes classifier based on each local table. The parameter $k = 3$ and the Gini index as a splitting criterion when building decision

Table 1. Results of Prec., Recall, F-m., *bacc* and *acc* for the global neural network.

Data set	No. tables	No. artif. obj.	One hidden layer					Two hidden layer				
			Prec.	Recall	F-m.	*bacc*	*acc*	Prec.	Recall	F-m.	*bacc*	*acc*
Vehicle imba-lanced	3	1	0.707	0.627	0.599	0.627	0.627	0.718	0.713	0.696	0.685	0.713
		2	0.671	0.681	0.656	0.655	0.681	0.703	0.715	0.701	0.68	0.715
		3	0.665	0.665	0.665	0.665	0.665	0.711	0.713	0.705	0.684	0.713
		4	0.685	0.68	0.665	0.651	0.68	0.738	0.74	0.737	0.721	**0.74**
		5	0.69	0.685	0.669	0.656	0.685	0.726	0.739	0.723	0.705	0.739
	5	1	0.685	0.673	0.658	0.646	0.673	0.713	0.715	0.707	0.692	0.715
		2	0.682	0.689	0.664	0.653	0.689	0.727	0.734	0.724	0.714	**0.734**
		3	0.669	0.671	0.645	0.639	0.671	0.711	0.72	0.714	0.694	0.72
		4	0.647	0.672	0.641	0.631	0.672	0.741	0.732	0.731	0.714	0.732
		5	0.713	0.689	0.655	0.65	0.689	0.695	0.702	0.693	0.67	0.702
	7	1	0.653	0.682	0.65	0.645	0.682	0.735	0.735	0.727	0.708	**0.735**
		2	0.701	0.697	0.674	0.669	0.697	0.728	0.717	0.711	0.691	0.717
		3	0.703	0.696	0.672	0.669	0.696	0.732	0.724	0.722	0.704	0.724
		4	0.701	0.706	0.688	0.682	0.706	0.725	0.73	0.72	0.701	0.73
		5	0.698	0.707	0.689	0.687	0.707	0.711	0.72	0.713	0.691	0.72
	9	1	0.705	0.697	0.689	0.674	0.697	0.746	0.752	0.739	0.719	**0.752**
		2	0.669	0.678	0.662	0.656	0.678	0.728	0.74	0.727	0.706	0.74
		3	0.706	0.717	0.699	0.688	0.717	0.713	0.723	0.708	0.687	0.723
		4	0.683	0.686	0.661	0.663	0.686	0.737	0.744	0.735	0.718	0.744
		5	0.712	0.703	0.681	0.671	0.703	0.733	0.734	0.723	0.704	0.734
	11	1	0.675	0.694	0.672	0.666	0.694	0.74	0.743	0.732	0.717	0.743
		2	0.709	0.714	0.698	0.69	0.714	0.752	0.756	0.744	0.726	**0.756**
		3	0.673	0.688	0.671	0.659	0.688	0.726	0.739	0.718	0.703	0.739
		4	0.658	0.664	0.641	0.634	0.664	0.75	0.744	0.736	0.724	0.744
		5	0.712	0.694	0.678	0.665	0.694	0.73	0.735	0.726	0.711	0.735
Vehicle ba-lanced	3	1	0.69	0.703	0.678	0.678	0.703	0.722	0.726	0.713	0.702	0.726
		2	0.714	0.685	0.664	0.67	0.685	0.713	0.731	0.717	0.699	0.731
		3	0.687	0.677	0.669	0.654	0.677	0.746	0.743	0.734	0.712	0.743
		4	0.693	0.697	0.684	0.676	0.697	0.751	0.761	0.741	0.728	0.761
		5	0.694	0.703	0.683	0.665	0.703	0.756	0.762	0.753	0.736	**0.762**
	5	1	0.699	0.71	0.689	0.674	0.71	0.748	0.748	0.73	0.725	0.748
		2	0.704	0.72	0.698	0.682	0.72	0.728	0.714	0.705	0.694	0.714
		3	0.692	0.717	0.696	0.684	0.717	0.734	0.734	0.721	0.701	0.734
		4	0.733	0.714	0.671	0.685	0.714	0.749	0.749	0.742	0.725	**0.749**
		5	0.686	0.703	0.682	0.675	0.703	0.723	0.739	0.717	0.702	0.739
	7	1	0.715	0.723	0.708	0.692	0.723	0.751	0.759	0.749	0.732	0.759
		2	0.715	0.726	0.708	0.691	0.726	0.755	0.753	0.75	0.731	0.753
		3	0.724	0.735	0.72	0.7	0.735	0.755	0.765	0.753	0.732	0.765
		4	0.713	0.724	0.684	0.679	0.724	0.767	0.756	0.751	0.734	0.756
		5	0.732	0.724	0.711	0.7	0.724	0.769	0.77	0.758	0.741	**0.77**
	9	1	0.74	0.743	0.732	0.715	0.743	0.744	0.745	0.743	0.721	0.745
		2	0.704	0.724	0.703	0.689	0.724	0.718	0.72	0.71	0.692	0.72
		3	0.707	0.711	0.695	0.681	0.711	0.743	0.741	0.739	0.718	0.741
		4	0.699	0.714	0.701	0.682	0.714	0.744	0.757	0.743	0.723	0.757
		5	0.728	0.722	0.704	0.693	0.722	0.757	0.76	0.747	0.732	**0.76**
	11	1	0.706	0.714	0.701	0.681	0.714	0.768	0.768	0.762	0.746	**0.768**
		2	0.709	0.713	0.704	0.68	0.713	0.702	0.722	0.694	0.684	0.722
		3	0.687	0.705	0.689	0.667	0.705	0.75	0.747	0.742	0.727	0.747
		4	0.71	0.71	0.705	0.682	0.71	0.723	0.73	0.721	0.698	0.73
		5	0.726	0.738	0.719	0.703	0.738	0.749	0.755	0.745	0.726	0.755

trees were used. The final decision of the ensemble was also made by soft voting. Both approaches are implemented in the Python programming language using implementations available in the sklearn library.

The results obtained for the proposed approach and the two approaches discussed above are given in Table 4. The best obtained F-measure, balanced accuracy and accuracy values for each data set and dispersed version are presented

Table 2. Results of Prec., Recall, F-m., *bacc* and *acc* for the global neural network.

Data set	No. tables	No. artif. obj.	One hidden layer					Two hidden layer				
			Prec.	Recall	F-m.	*bacc*	*acc*	Prec.	Recall	F-m.	*bacc*	*acc*
Satellite imbalanced	3	1	0.8	0.795	0.783	0.762	0.795	0.806	0.818	0.803	0.775	0.818
		2	0.798	0.796	0.778	0.759	0.796	0.82	0.822	0.818	0.798	0.822
		3	0.796	0.792	0.78	0.759	0.792	0.822	0.829	0.822	0.8	**0.829**
		4	0.782	0.799	0.778	0.758	0.799	0.821	0.828	0.822	0.798	0.828
		5	0.795	0.799	0.786	0.763	0.799	0.819	0.822	0.815	0.792	0.822
	5	1	0.806	0.81	0.798	0.773	0.81	0.818	0.824	0.818	0.795	0.824
		2	0.806	0.805	0.787	0.763	0.805	0.827	0.829	0.826	0.803	0.829
		3	0.789	0.801	0.781	0.761	0.801	0.831	0.833	0.829	0.803	0.833
		4	0.791	0.802	0.785	0.763	0.802	0.836	0.841	0.834	0.809	**0.841**
		5	0.793	0.795	0.779	0.757	0.795	0.832	0.839	0.831	0.803	0.839
	7	1	0.799	0.805	0.792	0.77	0.805	0.817	0.821	0.814	0.792	0.821
		2	0.814	0.809	0.798	0.776	0.809	0.831	0.838	0.831	0.807	0.838
		3	0.811	0.804	0.786	0.767	0.804	0.831	0.84	0.832	0.81	**0.84**
		4	0.8	0.804	0.791	0.768	0.804	0.826	0.833	0.824	0.793	0.833
		5	0.802	0.793	0.777	0.759	0.793	0.837	0.839	0.834	0.808	0.839
	9	1	0.8	0.808	0.799	0.775	0.808	0.831	0.832	0.825	0.803	0.832
		2	0.813	0.81	0.791	0.769	0.81	0.825	0.831	0.821	0.797	0.831
		3	0.806	0.813	0.8	0.778	0.813	0.835	0.839	0.827	0.8	**0.839**
		4	0.806	0.808	0.79	0.766	0.808	0.831	0.836	0.829	0.804	0.836
		5	0.818	0.804	0.781	0.759	0.804	0.834	0.839	0.834	0.807	**0.839**
	11	1	0.808	0.813	0.802	0.777	0.813	0.833	0.837	0.832	0.807	0.837
		2	0.815	0.808	0.792	0.77	0.808	0.832	0.839	0.832	0.809	**0.839**
		3	0.817	0.812	0.793	0.769	0.812	0.825	0.833	0.823	0.796	0.833
		4	0.805	0.81	0.796	0.772	0.81	0.825	0.835	0.825	0.798	0.835
		5	0.8	0.808	0.791	0.769	0.808	0.832	0.835	0.827	0.801	0.835
Satellite balanced	3	1	0.734	0.78	0.749	0.712	0.78	0.799	0.806	0.791	0.758	0.806
		2	0.753	0.77	0.747	0.721	0.77	0.802	0.803	0.797	0.768	0.803
		3	0.787	0.773	0.765	0.745	0.773	0.793	0.8	0.792	0.762	0.8
		4	0.773	0.772	0.758	0.734	0.772	0.805	0.809	0.802	0.776	**0.809**
		5	0.784	0.783	0.774	0.748	0.783	0.807	0.808	0.799	0.772	0.808
	5	1	0.771	0.776	0.766	0.732	0.776	0.812	0.811	0.803	0.776	0.811
		2	0.766	0.777	0.755	0.723	0.777	0.805	0.808	0.798	0.765	0.808
		3	0.791	0.791	0.784	0.757	0.791	0.796	0.803	0.789	0.759	0.803
		4	0.801	0.79	0.776	0.747	0.79	0.798	0.803	0.796	0.769	0.803
		5	0.794	0.791	0.78	0.755	0.791	0.807	0.814	0.806	0.775	**0.814**
	7	1	0.796	0.79	0.781	0.753	0.79	0.804	0.807	0.793	0.763	0.807
		2	0.773	0.791	0.768	0.739	0.791	0.808	0.814	0.803	0.774	**0.814**
		3	0.771	0.785	0.768	0.738	0.785	0.812	0.814	0.81	0.784	**0.814**
		4	0.811	0.791	0.771	0.749	0.791	0.807	0.813	0.806	0.778	0.813
		5	0.789	0.78	0.772	0.746	0.78	0.809	0.814	0.807	0.778	**0.814**
	9	1	0.778	0.788	0.764	0.734	0.788	0.818	0.822	0.813	0.786	**0.822**
		2	0.786	0.786	0.775	0.745	0.786	0.808	0.81	0.8	0.77	0.81
		3	0.782	0.78	0.764	0.737	0.78	0.805	0.815	0.803	0.773	0.815
		4	0.787	0.787	0.771	0.742	0.787	0.812	0.819	0.807	0.775	0.819
		5	0.781	0.792	0.771	0.742	0.792	0.803	0.81	0.801	0.769	0.81
	11	1	0.79	0.798	0.785	0.756	0.798	0.812	0.815	0.809	0.783	**0.815**
		2	0.778	0.791	0.773	0.739	0.791	0.803	0.81	0.803	0.773	0.81
		3	0.801	0.795	0.773	0.745	0.795	0.795	0.805	0.795	0.758	0.805
		4	0.79	0.788	0.775	0.745	0.788	0.807	0.812	0.804	0.779	0.812
		5	0.782	0.791	0.774	0.744	0.791	0.797	0.812	0.8	0.77	0.812

in bold in the table. These three measures were chosen for analysis as F-measure and balanced accuracy best illustrate the model's overall ability to correctly identify all decision classes and balance between precision and recall. The accuracy measure was also compared, but it is less significant in general, as it can lead to incorrect conclusions in the case of imbalanced data. As can be seen in the vast majority of cases, the proposed approach gives better results for all three

Table 3. Results of Prec., Recall, F-m., *bacc* and *acc* for the global neural network.

Data set	No. tables	No. artif. obj.	One hidden layer					Two hidden layer				
			Prec.	Recall	F-m.	*bacc*	*acc*	Prec.	Recall	F-m.	*bacc*	*acc*
Dry Bean imba-lanced	3	1	0.913	0.912	0.912	0.923	0.912	0.921	0.92	0.92	0.931	0.92
		2	0.918	0.917	0.917	0.927	0.917	0.921	0.921	0.921	0.931	**0.921**
		3	0.912	0.911	0.911	0.921	0.911	0.92	0.919	0.919	0.93	0.919
		4	0.914	0.913	0.913	0.922	0.913	0.92	0.92	0.92	0.93	0.92
		5	0.915	0.914	0.914	0.925	0.914	0.919	0.919	0.919	0.93	0.919
	5	1	0.912	0.912	0.912	0.922	0.912	0.918	0.918	0.918	0.928	0.918
		2	0.916	0.915	0.915	0.925	0.915	0.92	0.92	0.919	0.93	**0.92**
		3	0.914	0.913	0.913	0.923	0.913	0.918	0.917	0.917	0.927	0.917
		4	0.914	0.914	0.914	0.924	0.914	0.918	0.918	0.917	0.927	0.918
		5	0.915	0.915	0.915	0.925	0.915	0.918	0.918	0.918	0.928	0.918
	7	1	0.914	0.914	0.913	0.923	0.914	0.917	0.917	0.917	0.928	0.917
		2	0.914	0.914	0.914	0.924	0.914	0.919	0.919	0.919	0.93	**0.919**
		3	0.914	0.913	0.913	0.923	0.913	0.915	0.915	0.915	0.925	0.915
		4	0.913	0.913	0.913	0.923	0.913	0.916	0.916	0.916	0.927	0.916
		5	0.913	0.913	0.913	0.922	0.913	0.916	0.915	0.915	0.924	0.915
	9	1	0.915	0.914	0.914	0.924	0.914	0.916	0.915	0.915	0.925	0.915
		2	0.914	0.913	0.913	0.923	0.913	0.918	0.918	0.918	0.929	**0.918**
		3	0.914	0.913	0.913	0.923	0.913	0.915	0.915	0.915	0.924	0.915
		4	0.913	0.912	0.912	0.922	0.912	0.914	0.913	0.913	0.922	0.913
		5	0.913	0.912	0.912	0.921	0.912	0.915	0.915	0.915	0.924	0.915
	11	1	0.914	0.913	0.913	0.922	0.913	0.915	0.914	0.914	0.924	0.914
		2	0.914	0.914	0.914	0.924	0.914	0.918	0.918	0.918	0.928	**0.918**
		3	0.912	0.911	0.911	0.92	0.911	0.909	0.908	0.908	0.916	0.908
		4	0.913	0.912	0.912	0.922	0.912	0.912	0.912	0.912	0.92	0.912
		5	0.913	0.913	0.913	0.922	0.913	0.91	0.91	0.91	0.918	0.91
Dry Bean ba-lanced	3	1	0.913	0.912	0.912	0.922	0.912	0.92	0.919	0.919	0.93	0.919
		2	0.916	0.915	0.915	0.927	0.915	0.924	0.923	0.923	0.935	**0.923**
		3	0.912	0.911	0.911	0.921	0.911	0.92	0.92	0.919	0.931	0.92
		4	0.916	0.916	0.916	0.926	0.916	0.92	0.919	0.919	0.929	0.919
		5	0.914	0.914	0.914	0.923	0.914	0.92	0.92	0.919	0.929	0.92
	5	1	0.916	0.916	0.916	0.926	0.916	0.918	0.917	0.917	0.927	0.917
		2	0.917	0.916	0.916	0.928	0.916	0.921	0.921	0.921	0.932	**0.921**
		3	0.914	0.913	0.913	0.923	0.913	0.92	0.919	0.919	0.929	0.919
		4	0.914	0.913	0.913	0.923	0.913	0.918	0.918	0.918	0.929	0.918
		5	0.918	0.917	0.917	0.927	0.917	0.919	0.919	0.919	0.928	0.919
	7	1	0.913	0.912	0.911	0.922	0.912	0.919	0.919	0.919	0.93	**0.919**
		2	0.914	0.914	0.914	0.923	0.914	0.918	0.918	0.918	0.928	0.918
		3	0.914	0.913	0.913	0.922	0.913	0.918	0.918	0.917	0.927	0.918
		4	0.913	0.912	0.912	0.921	0.912	0.919	0.918	0.918	0.928	0.918
		5	0.914	0.913	0.913	0.922	0.913	0.918	0.918	0.917	0.927	0.918
	9	1	0.913	0.913	0.913	0.922	0.913	0.917	0.917	0.916	0.926	0.917
		2	0.908	0.907	0.907	0.915	0.907	0.919	0.919	0.919	0.929	**0.919**
		3	0.912	0.912	0.912	0.921	0.912	0.916	0.916	0.916	0.925	0.916
		4	0.914	0.913	0.913	0.923	0.913	0.917	0.917	0.917	0.926	0.917
		5	0.913	0.912	0.912	0.921	0.912	0.918	0.917	0.917	0.926	0.917
	11	1	0.912	0.912	0.912	0.922	0.912	0.917	0.917	0.917	0.926	0.917
		2	0.91	0.909	0.909	0.918	0.909	0.916	0.915	0.914	0.925	0.915
		3	0.913	0.912	0.912	0.921	0.912	0.92	0.92	0.919	0.929	**0.92**
		4	0.912	0.911	0.911	0.92	0.911	0.917	0.916	0.916	0.926	0.916
		5	0.913	0.913	0.913	0.922	0.913	0.914	0.914	0.913	0.922	0.914

compared measures. However, in the case of the Satellite data set, the proposed approach does not perform well. This is due to the data having the greatest variation in attributes present in local tables(very few overlapping attributes). To conclude, when a camera points at an object and generates attributes the majority of are not present in any other local tables, then the proposed approach does not perform better than the classifier ensemble approach. However, in other cases, when the cameras are more densely arranged, overlapping in terms of attributes then the proposed approach definitely performs better. It should also be noted that the number of objects in the training set does not affect the quality of classification generated by the proposed approach, i.e. for both small and large training sets the proposed approach gives good results.

Now, the results for all three measures generated by the analyzed approaches will be compared. To prove that the obtained differences in F-measure values are significant, the Friedman test was performed. Three dependent samples of 30 observations was used, with the test confirming that there is a statistically significant difference in the F-measure obtained for the three approaches considered, $\chi^2(29, 2) = 10.034, p = 0.007$. Additionally, comparative box-plot for the F-measure with three methods was created (Figure 2). As can be observed, on average, the values of the F-measure for the proposed approach are the largest. The post-hoc Dunn Bonferroni test was also performed which confirmed a significant difference in average F-measure values between the three approaches. The results (significant were presented in bold) can be found in Table 5. In the end, it can be said that the proposed approach improves the quality of classification compared to approaches known from the literature in terms of the F-measure.

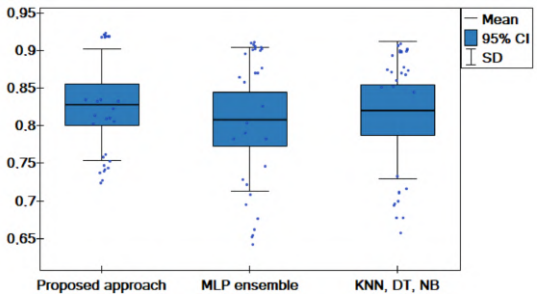


Fig. 2. Comparison of F-measure obtained for approaches: the proposed approach with global model; homogeneous ensemble with MLP networks (MLP ensemble) and the ensemble of classifiers: $k-$nearest neighbors, decision tree and naive bayes classifier (KNN, DT, NB).

Next the Friedman test was performed in order to show that the obtained differences in balanced accuracy values are significant. For balanced accuracy, the Friedman statistics was $7.983, p = 0.018$ indicating that there is a statistically significant difference in the balanced accuracy obtained for the three approaches

Table 4. Results of Prec., Recall, F-m., *bacc* and *acc* for the global neural network; homogeneous ensemble with MLP networks (MLP ensemble) and the ensemble of classifiers (KNN, DT, NB).

Data set	No. tables	The proposed approach				MLP ensemble				KNN, DT, NB			
		Prec.	Recall	F-m.	*bacc*	*acc*	Prec.	Recall	F-m.	*bacc*	*acc*	Prec.	Recall	F-m.	*bacc*	*acc*
Vehicle imbalanced	3	0.738	0.74	**0.737**	**0.721**	**0.74**	0.725	0.74	0.728	0.715	**0.74**	0.701	0.709	0.694	0.686	0.709
	5	0.727	0.734	**0.724**	0.714	**0.734**	0.761	0.724	0.695	**0.722**	0.724	0.718	0.709	0.696	0.69	0.709
	7	0.735	0.735	**0.727**	0.708	0.735	0.73	0.74	0.722	**0.709**	**0.74**	0.69	0.693	0.677	0.67	0.693
	9	0.746	0.752	**0.739**	**0.719**	**0.752**	0.655	0.685	0.652	0.658	0.685	0.708	0.717	0.699	0.694	0.717
	11	0.752	0.756	**0.744**	**0.726**	**0.756**	0.67	0.685	0.676	0.66	0.685	0.705	0.685	0.677	0.672	0.685
Vehicle balanced	3	0.756	0.762	**0.753**	**0.736**	**0.762**	0.742	0.756	0.746	0.729	0.756	0.733	0.732	0.716	0.705	0.732
	5	0.749	0.749	**0.742**	**0.725**	**0.749**	0.582	0.717	0.642	0.661	0.717	0.726	0.728	0.711	0.698	0.728
	7	0.769	0.77	**0.758**	**0.741**	**0.77**	0.711	0.728	0.708	0.699	0.728	0.748	0.752	0.733	0.721	0.752
	9	0.757	0.76	**0.747**	**0.732**	**0.76**	0.66	0.689	0.654	0.649	0.689	0.718	0.728	0.712	0.7	0.728
	11	0.768	0.768	**0.762**	**0.746**	**0.768**	0.683	0.685	0.662	0.656	0.685	0.667	0.677	0.657	0.646	0.677
Satellite imbalanced	3	0.822	0.829	0.822	0.8	0.829	0.838	0.849	0.826	0.8	0.849	0.868	0.87	**0.868**	**0.848**	**0.87**
	5	0.836	0.841	0.834	0.809	0.841	0.843	0.842	0.804	0.783	0.842	0.863	0.864	**0.86**	**0.835**	**0.864**
	7	0.831	0.84	0.832	0.81	0.84	0.758	0.835	0.79	0.768	0.835	0.855	0.857	**0.852**	**0.823**	**0.857**
	9	0.834	0.839	0.834	0.807	0.839	0.75	0.829	0.783	0.758	0.829	0.856	0.858	**0.851**	**0.82**	**0.858**
	11	0.832	0.839	0.832	0.809	0.839	0.754	0.828	0.783	0.757	0.828	0.851	0.854	**0.844**	**0.811**	**0.854**
Satellite balanced	3	0.805	0.809	0.802	0.776	0.809	0.865	0.868	0.864	0.839	0.868	0.879	0.872	**0.874**	**0.859**	**0.872**
	5	0.807	0.814	0.806	0.775	0.814	0.875	0.88	**0.876**	0.851	**0.88**	0.877	0.871	0.873	**0.856**	0.871
	7	0.812	0.814	0.81	0.784	0.814	0.873	0.869	0.87	0.859	0.869	0.881	0.878	**0.878**	**0.861**	**0.878**
	9	0.818	0.822	0.813	0.786	0.822	0.858	0.86	0.858	0.838	0.86	0.874	0.871	**0.871**	**0.851**	**0.871**
	11	0.812	0.815	0.809	0.783	0.815	0.873	0.87	**0.87**	**0.856**	**0.87**	0.871	0.87	**0.87**	0.848	**0.87**
Dry Bean imbalanced	3	0.921	0.921	**0.921**	**0.931**	**0.921**	0.911	0.91	0.91	0.92	0.91	0.908	0.906	0.906	0.916	0.906
	5	0.92	0.92	**0.919**	**0.93**	**0.92**	0.903	0.903	0.902	0.908	0.903	0.904	0.902	0.901	0.908	0.902
	7	0.919	0.919	**0.919**	**0.93**	**0.919**	0.902	0.901	0.9	0.906	0.901	0.901	0.899	0.899	0.905	0.899
	9	0.918	0.918	**0.918**	**0.929**	**0.918**	0.898	0.896	0.895	0.9	0.896	0.897	0.894	0.893	0.897	0.894
	11	0.918	0.918	**0.918**	**0.928**	**0.918**	0.902	0.901	0.901	0.907	0.901	0.902	0.9	0.9	0.905	0.9
Dry Bean balanced	3	0.924	0.923	**0.923**	**0.935**	**0.923**	0.911	0.911	0.911	0.926	0.911	0.91	0.909	0.909	0.919	0.909
	5	0.921	0.921	**0.921**	**0.932**	**0.921**	0.907	0.907	0.907	0.919	0.907	0.9	0.899	0.898	0.907	0.899
	7	0.919	0.919	**0.919**	**0.93**	**0.919**	0.906	0.905	0.905	0.917	0.905	0.901	0.9	0.899	0.909	0.9
	9	0.919	0.919	**0.919**	**0.929**	**0.919**	0.903	0.902	0.902	0.913	0.902	0.899	0.898	0.898	0.907	0.898
	11	0.92	0.92	**0.919**	**0.929**	**0.92**	0.905	0.904	0.904	0.916	0.904	0.903	0.903	0.902	0.912	0.903

considered in the paper. Additionally, comparative box-plot for the balanced accuracy was created (Figure 3). As can be seen also here, the average of balanced accuracy is the highest for the proposed approach. The post-hoc Dunn Bonferroni test confirmed a significant difference in balanced accuracy values between one pair: the proposed approach & MLP ensemble with $p = 0.035$. So it can be

Table 5. p-values for the post-hoc Dunn Bonferroni test for F-measure

	Proposed approach	MLP ensemble	KNN, DT, NB
Proposed approach	–	**0.011**	**0.035**
MLP ensemble	**0.011**	–	1
KNN, DT, NB	**0.035**	1	–

concluded that both the proposed approach and the ensemble of classifiers KNN, DT, NB get the best balanced accuracy results.

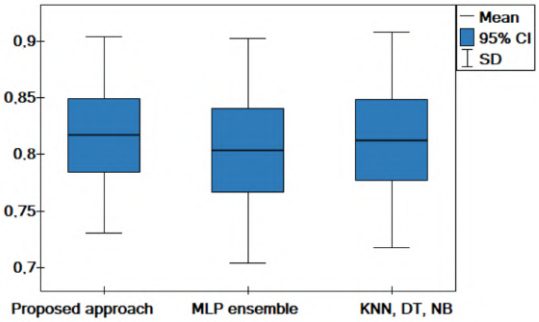


Fig. 3. Comparison of balanced accuracy obtained for approaches: the proposed approach with global model; homogeneous ensemble with MLP networks (MLP ensemble) and the ensemble of classifiers: $k-$nearest neighbors, decision tree and naive bayes classifier (KNN, DT, NB).

As we know, accuracy values can be deceptive and often do not take minority classes into account, nonetheless, comparative analysis was carried out for this measure. For accuracy, the Friedman statistics was $6.889, p = 0.032$ indicating a reject of the null hypothesis, but as can be seen in Figure 4, the average accuracy values are similar. Also, the post-hoc Dunn Bonferroni test did not confirm a significant difference between any pair of approaches. Thus, as a conclusion, it can be confirmed that the proposed approach on average improves values of F-measure and balanced accuracy. Of course, comparing the results obtained for each data set separately (Table 4), it can be seen that for some data sets this improvement is significant, while for others the proposed approach does not improve the quality of classification. The situation in which the proposed approach does not do well is when we have a very large number of conditional/descriptive attributes and a relatively small number of local tables/cameras.

Next, the impact of the number of artificial objects used in the proposed approach on the quality of classification is analyzed. The comparison, as before, was be made using three measures: F-measure, balanced accuracy and accuracy. To test whether the different number of artificial objects used generated a significant difference in results, five groups were created 1AO, 2AO, 3AO, 4AO, 5AO

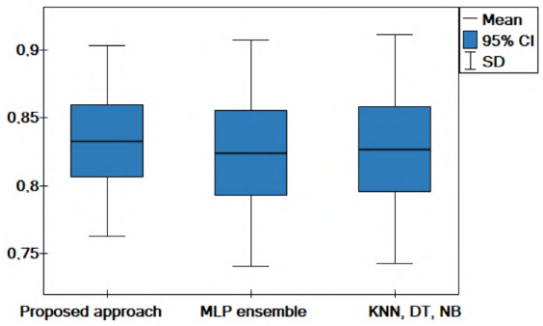


Fig. 4. Comparison of accuracy obtained for approaches: the proposed approach with global model; homogeneous ensemble with MLP networks (MLP ensemble) and the ensemble of classifiers: $k-$nearest neighbors, decision tree and naive bayes classifier (KNN, DT, NB).

– results obtained for 1, 2, 3, 4, 5 artificial objects used. Each group contained 60 observations (results obtained for all data sets and all versions of dispersion). The Friedman test confirmed a statistically significant difference in the F-measure obtained for the five groups considered, $\chi^2(59, 4) = 10.830, p = 0.029$. The Wilcoxon each-pair test confirmed the significant differences between the average F-measure values for the following pairs: 1AO & 4AO with $p = 0.01$, 1AO & 5AO with $p = 0.001$, 3AO & 4AO with $p = 0.005$, 3AO & 5AO with $p = 0.0002$. Additionally, a comparative graph for the F-measure with different number of artificial objects used was created (Figure 5). It can be seen that the results obtained for using 4 and 5 artificial objects are better than those obtained with fewer artificial objects.

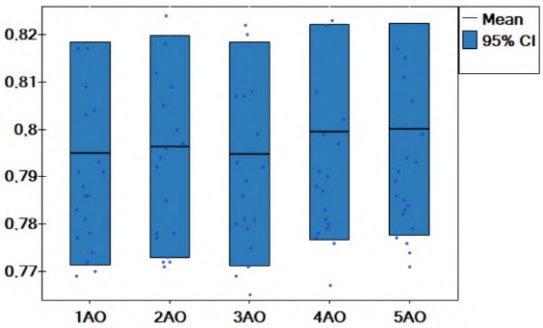


Fig. 5. Comparison of F-measure accuracy obtained for the proposed approach with 1, 2, 3, 4, 5 artificial objects generated (1AO, 2AO, 3AO, 4AO, 5AO).

Similar analyses were performed for balanced accuracy and accuracy. Friedman's test confirmed a significant difference for accuracy with statistics $15.501, p = 0.004$. The Wilcoxon each-pair test confirmed the significant differences between the average accuracy values for the following pairs: 1AO & 4AO with $p = 0.044$, 1AO & 5AO with $p = 0.002$, 2AO & 5AO with $p = 0.009$, 3AO & 4AO with $p = 0.002$, 3AO & 5AO with $p = 0.0001$. Also, the comparative graph for accuracy values (Figure 6) proves that for 5 and 4 artificial objects the generated results are better. For the balanced accuracy, the Friedman test does not confirm a significant difference in the mean value for different numbers of artificial objects used. Nonetheless, it can be concluded that larger numbers of artificial objects used to build the global model improve its quality. Thus, the proposed method of generating artificial objects with missing values in local tables has a positive effect on the model accuracy, and a larger number of artificial objects increases the quality of the model.

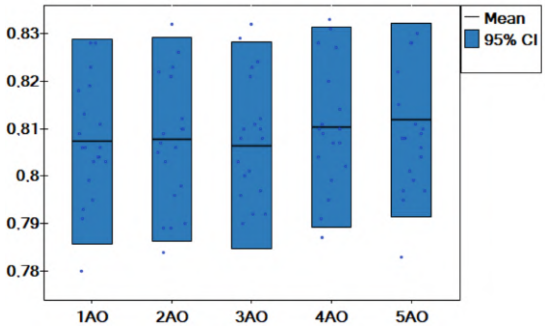


Fig. 6. Comparison of accuracy accuracy obtained for the proposed approach with 1, 2, 3, 4, 5 artificial objects generated (1AO, 2AO, 3AO, 4AO, 5AO).

4 Conclusion

In this paper, a situation in which local decision tables containing partial characteristics of objects from fragmented images was considered. Then, based on the local tables, tables with artificial objects were generated by filling the missing values of characteristics. Finally, local models were built based on local tables, which were finally aggregated into a global model.

In conclusion, it is important to note that while the proposed approach consistently enhances classification quality on average, its efficacy may vary across different data sets. The method excels particularly in scenarios with a substantial number of attributes and a relatively small number of local tables/cameras.

In summary, our study establishes the superiority of the proposed approach in terms of F-measure and balanced accuracy, showcasing its potential to elevate classification performance. The positive correlation between the number

of artificial objects and model quality reinforces the practical applicability of our method. These findings contribute valuable knowledge to the field of classification based on fragmented images, offering a promising avenue for further research and application in real-world scenarios.

References

1. Canal, F. Z., Müller, T. R., Matias, J. C., Scotton, G. G., de Sa Junior, A. R., Pozzebon, E., Sobieranski, A. C. (2022). A survey on facial emotion recognition techniques: A state-of-the-art literature review. Information Sciences, 582, 593-617.
2. Chaki, J., Dey, N., Moraru, L., Shi, F. (2019). Fragmented plant leaf recognition: Bag-of-features, fuzzy-color and edge-texture histogram descriptors with multi-layer perceptron. Optik, 181, 639-650.
3. Chen, J., Chen, J., Zhang, D., Sun, Y., Nanehkaran, Y. A. (2020). Using deep transfer learning for image-based plant disease identification. Computers and Electronics in Agriculture, 173, 105393.
4. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res., 16:321–357, 2002.
5. Çallı, E., Sogancioglu, E., van Ginneken, B., van Leeuwen, K. G., Murphy, K. (2021). Deep learning for chest X-ray analysis: A survey. Medical Image Analysis, 72, 102125.
6. Dua, D. and Graff, C.: UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, (2019)
7. Fornasier, M., Toniolo, D.: Fast, robust and efficient 2D pattern recognition for re-assembling fragmented images. Pattern Recognition, 38(11), 2074–2087, (2005)
8. Koklu, M., Ozkan, I. A.: Multiclass classification of dry beans using computer vision and machine learning techniques. Computers and Electronics in Agriculture, 174, 105507, (2020)
9. Lin, G., Tang, Y., Zou, X., Cheng, J., Xiong, J. (2020). Fruit detection in natural environment using partial shape matching and probabilistic Hough transform. Precision Agriculture, 21, 160-177.
10. Marfo, K. F., Przybyła-Kasperek, M.: Radial Basis Function Neural Network with a Centers Training Stage for Prediction Based on Dispersed Image Data. In International Conference on Computational Science (pp. 89-103). Cham: Springer Nature Switzerland, (2023, June)
11. Shelepin, Y. E., Chikhman, V. N., Foreman, N.: Analysis of the studies of the perception of fragmented images: global description and perception using local features. Neuroscience and behavioral physiology, 39(6), 569–580, (2009)
12. Siebert, J. P.: Vehicle Recognition Using Rule Based Methods, Turing Institute Research Memorandum TIRM-87-0.18, March 1987.
13. Vashist, P. C., Pandey, A., Tripathi, A. (2020, January). A comparative study of handwriting recognition techniques. In 2020 International Conference on Computation, Automation and Knowledge Management (ICCAKM) (pp. 456-461). IEEE.
14. Wu, L., Mokhtari, S., Nazef, A., Nam, B., Yun, H. B. (2016). Improvement of crack-detection accuracy using a novel crack defragmentation technique in image-based road assessment. Journal of Computing in Civil Engineering, 30(1), 04014118.
15. Xin, M., Wang, Y. (2019). Research on image classification model based on deep convolution neural network. EURASIP Journal on Image and Video Processing, 2019, 1-11.

Publication [P7]

Marfo K.F.,Przybyła-Kasperek M. Exploring the Impact of Object Diversity on Classification Quality in Dispersed Data Environments. *ACIIDS*, 14796:250-262, 2024.
URL: https://link.springer.com/chapter/10.1007/978-981-97-4985-0_20.
DOI: 10.1007/978-981-97-4985-0_20.
$\text{MEiN}_{2024} = 70$
Number of citations:

- according to Web of Science: 1
- according to Google Scholar: 2

Contributor	Description of main tasks
Kwabena Frimpong Marfo	- investigation - conceptualization and methodology - result analysis - implementation of proposed algorithm - manuscript: review and editing - visualization: creating all figures in manuscript
Małgorzata Przybyła-Kasperek	- conceptualization and methodology - result analysis - manuscript: original draft preparation - manuscript: review and editing - visualization: creating all figures in manuscript

Exploring the Impact of Object Diversity on Classification Quality in Dispersed Data Environments

Kwabena Frimpong Marfo[1][0000−0003−2226−9097] and Małgorzata Przybyła-Kasperek[1][0000−0003−0616−9694]

University of Silesia in Katowice, Institute of Computer Science, Będzińska 39, 41-200 Sosnowiec, Poland
{kmarfo,malgorzata.przybyla-kasperek}@us.edu.pl

Abstract. This paper studies a classification model dedicated to dispersed data, employing the $k-$nearest neighbors method as a base classifier and the Radial Basis Function (RBF) network as a fusion method. Focusing on classifying objects described by different attributes, the study systematically reduces common objects in local tables to assess the robustness of the model. Surprisingly, the proposed approach shows resilience in reducing common objects without significantly affecting key metrics such as F-measure, balanced accuracy and overall accuracy. Moreover, the studied model performs exceptionally well on imbalanced data. This research contributes valuable insights into dispersed data classification, demonstrating the model's effectiveness in handling diverse objects and attributes. The findings have implications for fields reliant on dispersed data storage, such as healthcare, banking, and surveillance, showcasing the model's potential for real-world applications.

Keywords: Dispersed data · Radial Basis Function network · Reduced common objects.

1 Introduction

In today's world, the presence of data-driven decision-making problems appears nearly on every facet of our lives. With the growth in data collection capabilities, the way of gathering data has gradually shifted towards dispersed data storage architectures. This approach entails the decentralized storage of data across various sources – a strategy employed in many fields such as healthcare, banking and surveillance. While dispersed data storage offers unparalleled advantages, it introduces a huge challenge – the harmonization of dispersed data with different features describing the same objects. Objects of interest, be they medical records, images captured from different viewpoints, or any other form of data are often represented in a fragmented manner across data silos, or sometimes, only a fraction of the features describing a given object is stored in a local table. This dispersion arises from the use of various sensors, data collection methods, or simply the decentralized nature of data source systems.

Fundamental to this challenge is the need to overcome the inconsistencies and differences in the way objects are represented locally. When there are different attributes and objects in local tables, achieving a uniform structure across different data sets is a non-trivial task. To address this challenge, this article proposes an approach that emphasizes on treating local data in isolation. Rather than imposing a global structure, the focus is on building local models that distinguish patterns in each data set independently. The synthesis of these local models, facilitated by neural networks, more specifically a RBF network, allows the identification of overarching patterns that transcend the local context. Neural networks, with their inherent ability to learn complex patterns and relationships are emerging as a promising tool in this respect.

The goal of this study is to investigate the possibility of building some general patterns that will provide good classification quality based on different objects described by a variety of attributes in local tables. To achieve this, a two-stage dispersion is considered in terms of both objects and attributes. In this study, we test the classification quality of a fusion model which comprises the $k-$nearest neighbor and a RBF network on dispersed data with diversified objects. We examine the results with gradually increasing object diversity and with dispersed attributes included in individual local tables.

The recognition of objects from fragmented data is a challenging domain with applications in various fields. Traditional approaches, as seen in [4, 11], have focused on assembling image fragments to form a complete representation. However, these methods assume non-overlapping fragments and often deal with visual images rather than tabular data. Diverse techniques have been explored for recognizing objects from fragmented or partial data. Among them are hidden Markov models, which were employed for gesture recognition based on fragmentary vision in [13], showcasing the adaptability of different methodologies to handle fragmented information. Also, fuzzy rules were utilized for fragmented handwritten digit recognition in [1], highlighting the versatility of rule-based systems in handling partial information. Addressing the challenges posed by data isolation and privacy concerns, Federated Learning (FL) has emerged as a privacy-preserving paradigm [6]. FL enables collaborative model training without the need to centralize sensitive data, making it particularly well-suited for scenarios involving dispersed data. The applicability of FL extends to fields such as healthcare, where patient records are dispersed across multiple hospitals, each maintaining its data silo. Existing fusion methods, as discussed in [10], employ k-nearest neighbors and Multi-Layer Perceptron (MLP) classifiers for dispersed data classification. While achieving favorable results, these methods do not scale well with noisy data. Perturbations in noise intensity and the number of neurons in the hidden layer significantly impact performance. Also, the above study considered homogeneous objects in local tables and did not account for the diversification of objects that could occur in local tables. In the present study, this is precisely the approach being considered.

The paper is organized as follows. In Section 2, the proposed classification model using a RBF neural network is described and the method for generating

dispersed data with a reduced number of common objects. Section 3 addresses the data sets that were used and presents the conducted experiments and discussion on obtained results. Section 4 is on conclusions and future research plans.

2 Materials and Methods

In the context of dispersed data storage, the process of constructing a unified structure of local data is a big challenge due to inherent data inconsistencies. To overcome this, an approach that treats local data in isolation is adopted. Patterns within each set are discovered by building local models, after which possible additional patterns in the predictions of the local models are sought out for using a neural network.

Consider a scenario where we have access to a local decision tables denoted as $D_i = (U_i, A_i, d)$ for $i \in 1, \ldots, n$. Here, U_i constitutes the universe, comprising a set of objects; A_i encompasses conditional attributes – features describing the objects; and d represents the decision attribute – labels. It is worth noting that objects and attributes in local tables may vary, with potential common elements being present among them.

This dispersion reflects situation in which independent sensors capture different features (some of them can be shared) on various objects in order to capture common patterns. It should be noted that the goal of these sensors is common – to make a decision on the same labels. In practical terms, such data can be found in dispersed medical data from many hospitals where one would want to make a prediction on the organ affected by cancer, or data from multiple cameras that capture objects from different perspectives in order to extract some general characteristics of the object. It is obvious that attributes differ in such an approach due to dispersion. In previous study [7, 8], a classification method based on dispersed data using the $k-$nearest neighbor classifier as the base classifier and a RBF neural network as the fusion method was proposed. This approach produced good results, however what it lacked was the diversification of objects, as objects in local tables were the same – only the attributes were dispersed. In the previous papers [7, 8], in order to model as close to a real situation as possible, object identifiers in local tables were not stored and recognized to generate global decisions, so aggregation of objects was not possible in any way. However, the question remains – whether uniformity/homogeneity of objects in local tables is necessary to achieve high quality in such a dispersed model. Perhaps, if the objects could also be different in local tables, one could analyze different patients in hospitals (which is very realistic) or show different object fragments to cameras and still retrieve patterns at the appropriate level of accuracy. In this study, we examine, for the first time, the effect of object diversity occurring in local tables on the obtained classification results. The robustness of the system against the presence of completely different objects between local tables is tested by gradually decreasing the percentage of occurring common objects between tables.

As mentioned in the papers [7, 8], local models are built based on local tables. Various models for building local classifiers can be used, however, in the previous research as well as in this paper, the k-nearest neighbors classifier was chosen for its low computational complexity and suitability for classification based on fragmented features. The expectation is for objects of the same type to exhibit similar features. To compute distances between test objects and object in local tables, the Gower measure is employed [9]. This measure can be used for attributes of different types (quantitative, qualitative, binary) and from different ranges in the decision table, eliminating the need for normalization or standardization. In the classification process, each base classifier performs classification using a subset of attributes that are present in the given local tables – a classifier i predicting on a decision table D_i utilizes the set of features A_i. It is crucial to note that the test objects must possess all features occurring in local tables . A classifier i produces a probability vector over decision classes for a test object x (denoted as $\mu_i(x)$). The vector's dimension is equal to the number of decision classes. Each coefficient $\mu_{i,j}(x)$ is determined using the $k-$nearest neighbors of the test object x belonging to a given decision class j and decision table D_i.

The challenging task of identifying the correct decision class for the test object is addressed using a RBF neural network. This approach has already yielded good results in previous papers [7, 8] and has proven successful in aggregating predictions generated from dispersed data when the same objects were stored in each local table, whereas the main issue analyzed now is the robustness of this model against the variation of objects in local tables. RBF networks have shown better classification quality than MLP networks so we focus on this approach in the paper. However, previous studies have not considered two-stage dispersion in terms of both objects and attributes, which is considered in this paper for the first time.

RBF network has the ability to transform input vectors into a new feature space where classes become linearly separable. For this transformation to be effective, the hidden layer must possess more neurons than the input layer. RBF neural networks inherently consist of only a hidden layer. In our scenario, the input vector is constructed from the prediction vectors generated by the base classifiers. Formally, a vector $[\mu_1(x), \ldots, \mu_n(x)]$ is created for the test object x, with a dimension of $n \cdot c$ since there are n base classifiers, each generating a c dimensional vector, where c is the number of decision classes. The input layer of the network consists of $n \cdot c$ neurons. Each neuron in the hidden layer is characterized by a parameter called the center. Formally, the k-th neuron in the hidden layer possesses the center c_k, interpreted as a representative of a specific group of objects (here it will be a specific group of prediction combinations). RBF networks reduces the influence of distant input on the network's output while incorporating them into the classification process. The back-propagation algorithm is employed to train the networks. In the experimental part, a stratified 10-fold cross-validation method is employed to train the RBF network on the test set. In this approach, the test set is partitioned into 10 folds, each containing an equal number of objects and maintaining proportional distributions

of decision classes. During each iteration, the network is trained using 9 folds of prediction vectors and the model's performance is assessed on the remaining fold. This process is repeated three times and the results are averaged to ascertain the model's accuracy. The most important impact studied in this paper is the division of objects among local tables so as to gradually reduce the percentage of common objects appearing in local tables. Initially, all training data used in the experimental part were part of a single table. Then attributes are dispersed among the local tables. However, a small part of the attributes are common between two or more tables. Objects are divided among local tables in accordance with Algorithm 1. In the first step the minimum number of objects each local table must have for there to be no common objects is calculated. The objects are then distributed in a stratified manner among local tables. A given p percentage of objects is selected from each local table. This drawn set of objects is then added to all other local tables. As can be seen, we take care of the stratified distribution of objects and that only p percent of the objects from a given local table are included in all other local decision tables. In the study, different percentages of common objects are checked, which are reduced by a step: 45%, 30% and 15%.

Algorithm 1 Algorithm of division objects between local tables

Input: A set of objects U with cardinality N,
p - percentage of diversification, percentage of common objects,
k - number of tables.
Output: Universe of local tables, $U_i, i \in \{1, \ldots, k\}$.

numOfLocalObj = $\left\lfloor \frac{N}{k} \right\rfloor$;
remainingObj = $N - (k \cdot \text{numOfLocalObj})$;
We divide objects from U in a stratified and disjoint manner into k subsets U_i each containing numOfLocalObj;
Objects from the set U that are not included in any local set U_i are added to all local sets U_i, their number is equal to remainingObj;
for each $i = 1 \ldots k$ do
$\rightarrow$ Choose p percent of the object from the set U_i;
$\rightarrow$ Save in S_i;
for each $i = 1 \ldots k$ do
$\rightarrow$ $U_i = U_i \cup \bigcup_{j \in \{1,\ldots,k\}, j \neq i} S_j$
Return $U_i, i \in \{1, \ldots, k\}$

3 Data sets and results

The experimental evaluation of the proposed system involves testing across three distinct data sets obtained from the UC Irvine Machine Learning Repository [3, 5, 12]. Let's delve into the specifics of each data set.

The Vehicle Silhouettes data set seeks to classify vehicle shapes into 4 categories by leveraging features extracted from images captured at various angles. This data set is characterized by 18 quantitative attributes, 4 decision classes, and a total of 846 objects. In the Landsat Satellite data set the task is to classify Earth types in satellite images based on multi-spectral pixel values within a 3×3 neighborhood. This data set has 36 quantitative attributes, 6 decision classes, and a substantial 6,435 objects. Lastly, the Dry Bean data set focuses on the classification of bean types, employing characteristics extracted from high-resolution images subjected to segmentation and feature extraction processes. It comprises 17 quantitative attributes, 7 decision classes, and an extensive 13,611 objects.

As part of the data pre-processing phase, a strategy of random dispersion of attributes and objects into 3, 5, 7, 9, and 11 local tables and different degree of percentage of common objects: small – 15%, medium – 30% and large – 45% is examined. The method of generating dispersed decision tables is consistent with that described in the previous section and presented in Algorithm 1. All the datasets used are imbalanced, with varying object counts across decision classes in both training and test sets. To explore the impact of imbalance on the proposed method and occurrence of common objects, two variants are considered for each data set: experiments on dispersed imbalanced data and on dispersed balanced data modified generated by the use of the Synthetic Minority Oversampling Technique (SMOTE) method [2].

The evaluation of classification performance is conducted on the test set, employing different measures. These measures include the Classification Accuracy (*acc*), which represents the fraction of correctly classified objects in the test set. Additionally, Recall evaluates the classifier's proficiency in recognizing a given class, while Precision (Prec.) expresses the classifier's precision in avoiding mistakes during classification. The F-measure (F-m.) provides a balanced assessment by combining Precision and Recall through the formula $\text{F-measure } = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$. Finally, Balanced Accuracy (*bacc*) calculates the average Recall for all decision classes, ensuring an equitable consideration of classification accuracy across all classes. All these metrics are calculated using functions implemented in Python in the sklearn library and using weighted average option. That is, the individual metrics are counted for each decision class and their weighted average (weights are the fraction of objects in each decision class) are found. This approach is very useful as it takes into account the imbalance of decision classes.

The experiments are carried out according to the following scheme:

- Generating prediction vectors from local tables using the $k-$nearest neighbors classifier involves an exploration of three values for the k parameter across each data set: $k \in \{1, 5, 10\}$.
- Generating predictions for different numbers of neurons in the hidden layer of the RBF network. The number of neurons depends on the number of neurons in the input layer of the network (which is equal to the number of decision classes times the number of local tables). The experimented values were

$\{0.25, 0.5, 0.75, 1, 1.5, 1.75, 2, 2.5, 2.75, 3, 3.5, 3.75, 4, 4.5, 4.75, 5\}$ times the number of neurons in the hidden layer.

The following research questions are studied when comparing the experimental results. Whether the number of common objects among local tables affect the classification quality of the RBF networks. Whether data balancing affects the classification quality of the RBF networks. The results obtained using the RBF network as a fusion method for different values of the parameter k ($k = 1$ is Table 1, $k = 5$ is Table 2 and $k = 10$ is Table 3) different number of common objects among local tables (45%, 30%, 15%) and different versions of attribute dispersion (3, 5, 7, 9, 11 local tables) are shown in Tables 1, 2, 3. The tables show results obtained for one number of neurons in the hidden layer chosen form different tested numbers in terms of the best classification accuracy selected separately for each tested data set and dispersion version. In Tables 1, 2, 3 the average values of results obtained from a 10-fold cross-validation on the test set are given. This means that the RBF network is trained 10 times with 9 folds and tested on one remaining fold. In addition, each test is performed three times to ensure that the results are reliable and not distorted by the influence of randomness.

We start the comparative analysis by examining whether object diversification (percentage of common objects between tables) has a significant impact on the results obtained using RBF networks as a fusion method. Statistical tests are performed for the three main measures: F-measure, balanced accuracy and accuracy. For each measure three dependent samples (with 45%, 30% and 15% of common objects among local tables) of 90 observations is used. The Friedman test confirmed that there is a statistically significant difference only in the balanced accuracy obtained for the three approaches considered with $\chi^2(90, 2) = 7.058, p = 0.03$. For the accuracy and the F-measure, the Friedman test showed no significant difference in the average results obtained. In addition, the Wilcoxon of each pair test showed a significant difference for balanced accuracy only between the approaches containing 15% and 30% common objects with $p = 0.03$. Comparative box-plot for the F-measure, balanced accuracy and accuracy with three different percentages of common objects was created (Figure 1). As can be seen from the figure, the difference between the averages of measures is really small. So, it can be said that there is no difference in the results generated by the RBF networks as a fusion method for different degrees of common objects among the local tables. Thus, the method is resistant to the variation of objects in local tables. Ultimately, the data can be fragmented in terms of both objects and attributes.

We now examine the effect of balancing on the results obtained in terms of F-measure, balanced accuracy and accuracy. This time, two dependent samples for the F-measure, balanced accuracy and accuracy, each containing 135 observations obtained for imbalanced and balanced data are created. The Wilcoxon test confirmed significant differences in the averages with $p = 0.0001$ for all measures. Comparative box-plot for the F-measure, balanced accuracy and accuracy for imbalanced and balanced results was created (Figure 2). As can be seen from

Table 1. Results of Prec., Recall, F-m., *bacc* and *acc* for the proposed approach with different degrees of diversification and $k = 1$ in the nearest neighbor algorithm.

Data set	No. tables	Percent of common objects	Imbalanced					Balanced				
			Prec.	Recall	F-m.	*bacc*	*acc*	Prec.	Recall	F-m.	*bacc*	*acc*
Vehicle		45%	0.749	0.75	0.734	0.721	0.75	0.74	0.737	0.727	0.706	0.737
	3	30%	0.772	0.758	0.748	0.736	0.758	0.775	0.756	0.745	0.732	0.756
		15%	0.788	0.78	0.769	0.762	0.78	0.747	0.734	0.721	0.715	0.734
		45%	0.781	0.775	0.766	0.73	0.775	0.642	0.742	0.68	0.695	0.742
	5	30%	0.771	0.763	0.758	0.723	0.763	0.727	0.762	0.727	0.722	0.762
		15%	0.772	0.763	0.749	0.729	0.763	0.652	0.743	0.681	0.689	0.743
		45%	0.786	0.759	0.754	0.747	0.759	0.678	0.717	0.678	0.69	0.717
	7	30%	0.773	0.749	0.745	0.728	0.749	0.761	0.742	0.733	0.718	0.742
		15%	0.736	0.733	0.723	0.711	0.733	0.71	0.714	0.698	0.685	0.714
		45%	0.788	0.775	0.755	0.74	0.775	0.635	0.715	0.66	0.681	0.715
	9	30%	0.77	0.735	0.718	0.711	0.735	0.685	0.692	0.668	0.652	0.692
		15%	0.782	0.762	0.752	0.741	0.762	0.715	0.712	0.694	0.67	0.712
		45%	0.76	0.741	0.736	0.718	0.741	0.727	0.726	0.705	0.687	0.726
	11	30%	0.751	0.739	0.733	0.709	0.739	0.738	0.735	0.722	0.712	0.735
		15%	0.775	0.754	0.741	0.724	0.754	0.773	0.765	0.747	0.74	0.765
Satellite		45%	0.906	0.904	0.902	0.889	0.904	0.899	0.9	0.897	0.874	0.9
	3	30%	0.902	0.899	0.897	0.885	0.899	0.896	0.892	0.89	0.867	0.892
		15%	0.905	0.903	0.901	0.887	0.903	0.898	0.894	0.893	0.87	0.894
		45%	0.894	0.89	0.888	0.877	0.89	0.884	0.881	0.88	0.858	0.881
	5	30%	0.896	0.894	0.893	0.876	0.894	0.883	0.88	0.879	0.855	0.88
		15%	0.891	0.887	0.886	0.874	0.887	0.878	0.877	0.874	0.85	0.877
		45%	0.889	0.887	0.885	0.867	0.887	0.882	0.881	0.877	0.854	0.881
	7	30%	0.894	0.89	0.889	0.877	0.89	0.886	0.884	0.881	0.858	0.884
		15%	0.891	0.887	0.886	0.871	0.887	0.877	0.873	0.869	0.839	0.873
		45%	0.887	0.884	0.882	0.867	0.884	0.877	0.876	0.873	0.849	0.876
	9	30%	0.895	0.889	0.888	0.877	0.889	0.87	0.871	0.868	0.84	0.871
		15%	0.895	0.891	0.89	0.876	0.891	0.861	0.862	0.853	0.819	0.862
		45%	0.886	0.882	0.881	0.866	0.882	0.885	0.883	0.88	0.85	0.883
	11	30%	0.891	0.886	0.885	0.871	0.886	0.877	0.874	0.871	0.843	0.874
		15%	0.885	0.882	0.88	0.866	0.882	0.879	0.876	0.874	0.853	0.876
Dry Bean		45%	0.92	0.919	0.919	0.928	0.919	0.916	0.914	0.914	0.925	0.914
	3	30%	0.918	0.917	0.916	0.926	0.917	0.917	0.916	0.916	0.926	0.916
		15%	0.922	0.921	0.92	0.931	0.921	0.92	0.918	0.918	0.926	0.918
		45%	0.922	0.921	0.921	0.931	0.921	0.92	0.918	0.918	0.929	0.918
	5	30%	0.923	0.922	0.922	0.932	0.922	0.918	0.917	0.916	0.926	0.917
		15%	0.921	0.92	0.92	0.929	0.92	0.917	0.916	0.916	0.926	0.916
		45%	0.92	0.918	0.918	0.928	0.918	0.919	0.918	0.918	0.927	0.918
	7	30%	0.922	0.921	0.921	0.932	0.921	0.921	0.92	0.92	0.927	0.92
		15%	0.923	0.921	0.921	0.93	0.921	0.92	0.918	0.918	0.927	0.918
		45%	0.922	0.921	0.921	0.929	0.921	0.917	0.916	0.915	0.925	0.916
	9	30%	0.919	0.918	0.918	0.926	0.918	0.919	0.918	0.918	0.926	0.918
		15%	0.923	0.922	0.922	0.931	0.922	0.922	0.921	0.921	0.929	0.921
		45%	0.92	0.918	0.918	0.927	0.918	0.918	0.916	0.916	0.923	0.916
	11	30%	0.919	0.918	0.918	0.928	0.918	0.917	0.916	0.916	0.925	0.916
		15%	0.921	0.919	0.919	0.929	0.919	0.917	0.916	0.915	0.924	0.916

the figure, the differences between the averages of the F-measure, balanced accuracy and accuracy are small but visible. What may be surprising is that it is with imbalanced data sets that, on average, the results achieved are better. It can be concluded that the RBF network as a fusion method performs well with imbalanced data sets with even a small number of common objects between tables.

One more statistical analysis is performed to compare both the results in terms of the percentage of common objects in local tables and the imbalance of data sets. This time, six dependent samples are compared: 45%, 30%, 15%

Table 2. Results of Prec., Recall, F-m., *bacc* and *acc* for the proposed approach with different degrees of diversification and $k = 5$ in the nearest neighbor algorithm.

Data set	No. tables	Percent of common objects	Imbalanced					Balanced				
			Prec.	Recall	F-m.	*bacc*	*acc*	Prec.	Recall	F-m.	*bacc*	*acc*
Vehicle		45%	0.776	0.761	0.749	0.734	0.761	0.748	0.727	0.715	0.701	0.727
	3	30%	0.776	0.748	0.74	0.713	0.748	0.741	0.742	0.727	0.726	0.742
		15%	0.799	0.782	0.774	0.759	0.782	0.774	0.765	0.751	0.743	0.765
		45%	0.794	0.777	0.769	0.752	0.777	0.781	0.776	0.763	0.739	0.776
	5	30%	0.773	0.765	0.752	0.723	0.765	0.764	0.768	0.754	0.737	0.768
		15%	0.772	0.753	0.746	0.718	0.753	0.8	0.785	0.779	0.758	0.785
		45%	0.774	0.767	0.759	0.742	0.767	0.746	0.738	0.726	0.715	0.738
	7	30%	0.777	0.761	0.747	0.733	0.761	0.74	0.728	0.713	0.696	0.728
		15%	0.782	0.773	0.765	0.742	0.773	0.747	0.748	0.727	0.723	0.748
		45%	0.763	0.743	0.737	0.731	0.743	0.717	0.734	0.711	0.704	0.734
	9	30%	0.755	0.744	0.74	0.726	0.744	0.721	0.732	0.706	0.705	0.732
		15%	0.767	0.757	0.749	0.743	0.757	0.719	0.708	0.696	0.674	0.708
		45%	0.777	0.757	0.743	0.721	0.757	0.75	0.75	0.734	0.721	0.75
	11	30%	0.737	0.748	0.735	0.72	0.748	0.736	0.722	0.715	0.698	0.722
		15%	0.8	0.783	0.769	0.751	0.783	0.752	0.749	0.737	0.728	0.749
Satellite		45%	0.898	0.896	0.894	0.879	0.896	0.902	0.898	0.896	0.872	0.898
	3	30%	0.895	0.893	0.891	0.874	0.893	0.893	0.892	0.89	0.867	0.892
		15%	0.902	0.9	0.899	0.884	0.9	0.903	0.901	0.899	0.875	0.901
		45%	0.9	0.897	0.895	0.882	0.897	0.889	0.888	0.885	0.862	0.888
	5	30%	0.903	0.901	0.899	0.884	0.901	0.89	0.887	0.884	0.863	0.887
		15%	0.899	0.896	0.895	0.877	0.896	0.892	0.887	0.885	0.865	0.887
		45%	0.895	0.891	0.891	0.874	0.891	0.896	0.892	0.89	0.869	0.892
	7	30%	0.895	0.891	0.89	0.88	0.891	0.884	0.883	0.88	0.856	0.883
		15%	0.898	0.894	0.893	0.88	0.894	0.888	0.886	0.883	0.859	0.886
		45%	0.9	0.896	0.894	0.882	0.896	0.884	0.884	0.881	0.858	0.884
	9	30%	0.897	0.893	0.892	0.878	0.893	0.889	0.887	0.884	0.862	0.887
		15%	0.901	0.897	0.896	0.882	0.897	0.883	0.879	0.877	0.853	0.879
		45%	0.893	0.888	0.887	0.872	0.888	0.882	0.881	0.879	0.854	0.881
	11	30%	0.893	0.891	0.889	0.873	0.891	0.877	0.875	0.873	0.849	0.875
		15%	0.894	0.89	0.889	0.876	0.89	0.888	0.884	0.882	0.862	0.884
Dry Bean		45%	0.921	0.921	0.92	0.931	0.921	0.921	0.92	0.92	0.929	0.92
	3	30%	0.92	0.92	0.919	0.93	0.92	0.921	0.92	0.919	0.927	0.92
		15%	0.922	0.921	0.92	0.931	0.921	0.919	0.918	0.918	0.928	0.918
		45%	0.922	0.92	0.92	0.932	0.92	0.918	0.917	0.917	0.926	0.917
	5	30%	0.923	0.922	0.922	0.932	0.922	0.92	0.919	0.919	0.927	0.919
		15%	0.921	0.92	0.919	0.93	0.92	0.92	0.919	0.919	0.928	0.919
		45%	0.921	0.92	0.92	0.93	0.92	0.922	0.921	0.921	0.929	0.921
	7	30%	0.92	0.919	0.919	0.929	0.919	0.921	0.92	0.92	0.93	0.92
		15%	0.921	0.92	0.919	0.931	0.92	0.921	0.92	0.92	0.93	0.92
		45%	0.92	0.918	0.918	0.928	0.918	0.92	0.918	0.918	0.928	0.918
	9	30%	0.921	0.92	0.92	0.929	0.92	0.922	0.92	0.92	0.928	0.92
		15%	0.921	0.92	0.92	0.93	0.92	0.921	0.92	0.92	0.928	0.92
		45%	0.919	0.918	0.918	0.928	0.918	0.919	0.918	0.918	0.926	0.918
	11	30%	0.921	0.92	0.919	0.929	0.92	0.919	0.917	0.917	0.925	0.917
		15%	0.922	0.921	0.92	0.93	0.921	0.918	0.917	0.916	0.926	0.917

of common objects for imbalanced data sets and 45%, 30%, 15% of common objects for balanced data sets. In this way, we overcome the situation where data balance could disturb the overall conclusion regarding the effect of the number of common objects in the data set on the results. The Friedman test confirmed that there is a statistically significant difference in averages for all three measures with $\chi^2(45, 5) = 81.79, p = 0.000001$ for the F-measure, $\chi^2(45, 5) = 99.18, p = 0.000001$ for the balanced accuracy and $\chi^2(45, 5) = 75.46, p = 0.000001$ for the accuracy. In addition, the Wilcoxon of each pair test confirmed the significant difference in averages for all measures between the first three groups and the last

Table 3. Results of Prec., Recall, F-m., *bacc* and *acc* for the proposed approach with different degrees of diversification and $k = 10$ in the nearest neighbor algorithm.

Data set	No. tables	Percent of common objects	Imbalanced					Balanced				
			Prec.	Recall	F-m.	*bacc*	*acc*	Prec.	Recall	F-m.	*bacc*	*acc*
Vehicle		45%	0.774	0.751	0.735	0.713	0.751	0.746	0.747	0.734	0.725	0.747
	3	30%	0.776	0.741	0.731	0.715	0.741	0.746	0.736	0.717	0.71	0.736
		15%	0.744	0.74	0.734	0.707	0.74	0.768	0.767	0.756	0.743	0.767
		45%	0.802	0.784	0.775	0.749	0.784	0.782	0.777	0.763	0.746	0.777
	5	30%	0.765	0.756	0.743	0.722	0.756	0.786	0.771	0.766	0.749	0.771
		15%	0.758	0.738	0.726	0.705	0.738	0.777	0.762	0.756	0.736	0.762
		45%	0.792	0.782	0.778	0.758	0.782	0.763	0.755	0.741	0.723	0.755
	7	30%	0.744	0.75	0.73	0.709	0.75	0.758	0.745	0.733	0.718	0.745
		15%	0.763	0.754	0.743	0.721	0.754	0.737	0.743	0.724	0.717	0.743
		45%	0.762	0.753	0.747	0.736	0.753	0.737	0.732	0.714	0.705	0.732
	9	30%	0.753	0.741	0.733	0.719	0.741	0.726	0.735	0.714	0.708	0.735
		15%	0.76	0.755	0.746	0.731	0.755	0.712	0.711	0.694	0.683	0.711
		45%	0.779	0.772	0.757	0.734	0.772	0.737	0.743	0.722	0.704	0.743
	11	30%	0.759	0.753	0.745	0.725	0.753	0.724	0.729	0.714	0.692	0.729
		15%	0.765	0.764	0.749	0.73	0.764	0.755	0.748	0.738	0.72	0.748
Satellite		45%	0.895	0.892	0.89	0.875	0.892	0.895	0.893	0.891	0.869	0.893
	3	30%	0.896	0.893	0.892	0.877	0.893	0.898	0.897	0.894	0.872	0.897
		15%	0.903	0.901	0.899	0.881	0.901	0.896	0.895	0.893	0.87	0.895
		45%	0.896	0.893	0.892	0.877	0.893	0.889	0.886	0.884	0.861	0.886
	5	30%	0.9	0.896	0.895	0.881	0.896	0.893	0.89	0.888	0.867	0.89
		15%	0.895	0.891	0.89	0.875	0.891	0.895	0.892	0.89	0.87	0.892
		45%	0.896	0.89	0.89	0.878	0.89	0.893	0.891	0.889	0.867	0.891
	7	30%	0.897	0.893	0.892	0.878	0.893	0.887	0.885	0.883	0.86	0.885
		15%	0.897	0.892	0.891	0.878	0.892	0.889	0.885	0.882	0.86	0.885
		45%	0.894	0.891	0.89	0.878	0.891	0.892	0.889	0.887	0.867	0.889
	9	30%	0.899	0.895	0.894	0.879	0.895	0.885	0.881	0.879	0.857	0.881
		15%	0.897	0.894	0.892	0.876	0.894	0.887	0.883	0.88	0.858	0.883
		45%	0.896	0.892	0.891	0.875	0.892	0.883	0.883	0.879	0.855	0.883
	11	30%	0.896	0.892	0.891	0.878	0.892	0.878	0.877	0.873	0.85	0.877
		15%	0.891	0.888	0.886	0.871	0.888	0.885	0.88	0.877	0.859	0.88
Dry Bean		45%	0.921	0.92	0.92	0.931	0.92	0.921	0.92	0.92	0.929	0.92
	3	30%	0.923	0.922	0.922	0.931	0.922	0.922	0.921	0.921	0.931	0.921
		15%	0.922	0.921	0.921	0.931	0.921	0.918	0.917	0.917	0.926	0.917
		45%	0.923	0.922	0.922	0.931	0.922	0.92	0.919	0.918	0.928	0.919
	5	30%	0.922	0.921	0.921	0.931	0.921	0.921	0.919	0.919	0.927	0.919
		15%	0.923	0.922	0.922	0.932	0.922	0.921	0.92	0.919	0.928	0.92
		45%	0.922	0.92	0.92	0.931	0.92	0.918	0.917	0.917	0.927	0.917
	7	30%	0.92	0.919	0.919	0.927	0.919	0.92	0.919	0.918	0.927	0.919
		15%	0.92	0.919	0.919	0.93	0.919	0.919	0.919	0.918	0.927	0.919
		45%	0.919	0.918	0.918	0.927	0.918	0.919	0.918	0.918	0.927	0.918
	9	30%	0.92	0.919	0.919	0.928	0.919	0.918	0.917	0.917	0.927	0.917
		15%	0.92	0.919	0.919	0.928	0.919	0.921	0.919	0.919	0.928	0.919
		45%	0.921	0.919	0.919	0.928	0.919	0.92	0.918	0.918	0.928	0.918
	11	30%	0.919	0.918	0.918	0.927	0.918	0.92	0.919	0.919	0.927	0.919
		15%	0.923	0.922	0.922	0.932	0.922	0.92	0.919	0.919	0.929	0.919

three groups, so generally, imbalanced and balanced approaches. Comparative box-plot for the F-measure, balanced accuracy and accuracy for imbalanced and balanced data sets and 45%, 30%, 15% of common objects among local tables (Figure 3) confirms that only balancing of data sets affects results. The considered method is completely insensitive to the number of common objects present in local tables.

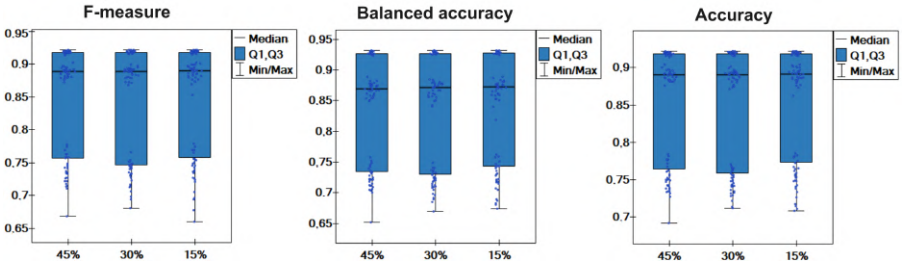


Fig. 1. Comparison of F-measure, balanced accuracy and accuracy obtained for the RBF networks as a fusion method for different degrees percentages of common objects among local tables 45%, 30% and 15%.

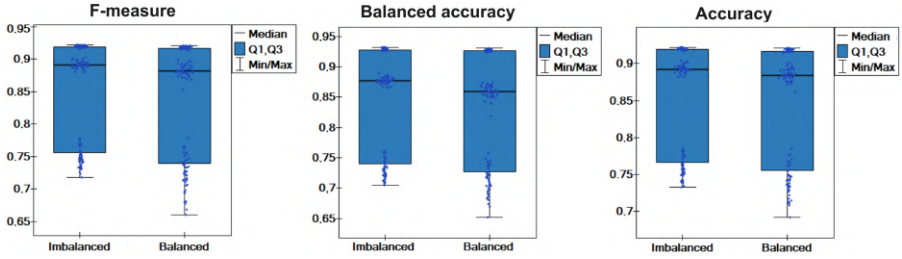


Fig. 2. Comparison of F-measure, balanced accuracy and accuracy obtained for the RBF networks as a fusion method for imbalanced and balanced data sets.

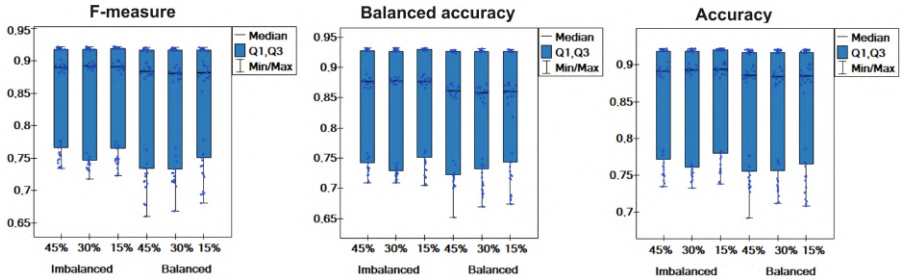


Fig. 3. Comparison of F-measure, balanced accuracy and accuracy obtained for the RBF networks as a fusion method for imbalanced and 45%, 30%, 15% of common objects among local tables and balanced data sets and 45%, 30%, 15% of common objects among local tables.

4 Conclusions

This paper studied a classification model based on dispersed data using the $k-$nearest neighbor classifier as the base classifier and the RBF network as the fusion method. It was investigated whether the dispersion of both objects and

attributes significantly affects the classification quality of the method. The study was conducted by gradually reducing the percentage of common attributes in local tables. The key conclusion of the study is that the proposed classification model is robust to a decreasing number of common objects in local tables. There is no significant effect on the average values of the measures like the F-measure, balanced accuracy and accuracy obtained for different percentages of common objects in local tables. In addition, it was noted that a key factor affecting the result is the balancing of the data set, and here it came out as a surprise that for dispersed data the proposed approach performs better for imbalanced data sets. In future work, it is planned to study the effect of the number of common conditional attributes on the quality of classification based on dispersed data.

References

1. Chaki, J., Dey, N.: Fragmented handwritten digit recognition using grading scheme and fuzzy rules. Sādhanā, 45(1), 1–23, (2020)
2. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res., 16:321–357, 2002.
3. Dua, D. and Graff, C.: UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, (2019)
4. Fornasier, M., Toniolo, D.: Fast, robust and efficient 2D pattern recognition for re-assembling fragmented images. Pattern Recognition, 38(11), 2074–2087, (2005)
5. Koklu, M., Ozkan, I. A.: Multiclass classification of dry beans using computer vision and machine learning techniques. Computers and Electronics in Agriculture, 174, 105507, (2020)
6. Li, L., Fan, Y., Tse, M., Lin, K. Y. (2020). A review of applications in federated learning. Computers & Industrial Engineering, 149, 106854.
7. Marfo, K. F., Przybyła-Kasperek, M. (2022). Radial basis function network for aggregating predictions of k-nearest neighbors local models generated based on independent data sets. Procedia Computer Science, 207, 3234-3243.
8. Marfo, K. F., Przybyła-Kasperek, M. (2023, June). Radial Basis Function Neural Network with a Centers Training Stage for Prediction Based on Dispersed Image Data. In International Conference on Computational Science (pp. 89-103). Cham: Springer Nature Switzerland.
9. Przybyła-Kasperek, M., Wakulicz-Deja, A. (2014). A dispersed decision-making system–The use of negotiations during the dynamic generation of a system's structure. Information Sciences, 288, 194-219.
10. Przybyła-Kasperek, M., Marfo, K. F.: Neural Network Used for the Fusion of Predictions Obtained by the K-Nearest Neighbors Algorithm Based on Independent Data Sources. Entropy 23, no. 12 (2021): 1568.
11. Shelepin, Y. E., Chikhman, V. N., Foreman, N.: Analysis of the studies of the perception of fragmented images: global description and perception using local features. Neuroscience and behavioral physiology, 39(6), 569–580, (2009)
12. Siebert, J. P.: Vehicle Recognition Using Rule Based Methods, Turing Institute Research Memorandum TIRM-87-0.18, March 1987.
13. Yang, R., Sarkar, S.: Gesture recognition using hidden markov models from fragmented observations. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06) vol. 1, 766–773. IEEE, (2006)

Publication [P8]

Marfo K.F.,Przybyła-Kasperek M. Enhancing Dispersed Data Classification: A Hierarchical Model Based on Neural Networks *Proceedings of the 5th Polish Conference on Artificial Intelligence*, 154-161, 2024.
URL: https://pages.mini.pw.edu.pl/ estatic/pliki/PP-RAI_2024_proceedings.pdf.
DOI: 10.17388/WUT.2024.0002.MiNI.
$\text{MEiN}_{2024} = 20$
Number of citations:

- according to Web of Science: 0
- according to Google Scholar: 0

Contributor	Description of main tasks
Kwabena Frimpong Marfo	- investigation - conceptualization and methodology - result analysis - implementation of proposed algorithm - manuscript: review and editing - visualization: creating all figures in manuscript
Małgorzata Przybyła-Kasperek	- conceptualization and methodology - result analysis - manuscript: original draft preparation - manuscript: review and editing - visualization: creating all figures in manuscript

Enhancing Dispersed Data Classification: A Hierarchical Model Based on Neural Networks

Kwabena Frimpong Marfo[1][0000−0003−2226−9097]
Małgorzata Przybyła-Kasperek[1][0000−0003−0616−9694]

[1]*University of Silesia in Katowice*
Institute of Computer Science,
Będzińska 39, 41-200 Sosnowiec, Poland
kwabena.marfo@us.edu.pl, malgorzata.przybyla-kasperek@us.edu.pl

Abstract. *The paper uses dispersed data, that is, data collected from independent sources. More specifically, dispersed tabular data is considered with the assumption that a set of local decision tables are available. In as much as each local table has attributes and objects unique to it, there may be attributes and objects that are common in more than a local table. The proposed approach uses a hierarchical model – at the lower level, local tables are transformed using a feature extraction method to generate tables with homogeneous structure. At the higher level, a neural network model is created based on such transformed local tables. Some advantages that the model presents are data protection (since raw data are not sent to the global model), the use of a global model despite the use of dispersed data and a reduced feature space – reducing model complexity. The paper also describes a classification method of new cases based on such a hierarchical model.*
Keywords: *Dispersed Data, Neural Networks, Principal Component Analysis, Multilayer Perceptron*

1. Introduction

The increasing volume and complexity of data in various domains and the limitations of centralized learning models resulted in exploration of dispersed data and dispersed learning paradigms. Perceptions on the use of dispersed data vary depending on context and approach.

Distributed learning involves training models across multiple nodes, each processing a subset of the data. This approach is particularly valuable in scenarios

DOI:10.17388/WUT.2024.0002.MiNI

where centralized processing is impractical due to the sheer volume of data or the need for real-time learning. Distributed learning algorithms have been explored to improve model training efficiency and scalability. Some of the well-known and effective approaches are: XGBoost [1], LightGBM [2], Random Forest [3] among others.

The approach discussed in this paper is also related to distributed learning. We assume that dispersed data collected independently in the form of local tables are available. Moreover, local tables do not satisfy any attribute or object constraints. This means that sets of attributes may be different between local tables, but some attributes may be common. In addition, the number of attributes between local tables can vary. Initially, the use of the k-nearest neighbor classifier as a base classifier for each local table separately was analyzed, and then the obtained prediction results were aggregated using neural networks. It should be mentioned that different types of neural networks were considered [4, 5, 6]. Next, an approach was analyzed in which neural networks with the same structure were built locally, using artificial objects. Such networks were then aggregated by averaging [7]. A completely different approach is now being considered, which aims to generate completely different local tables with an equal number of attributes based on the original tables. For this purpose, different approaches can be used as a transformation of the original data using a reduced number of components. Here we mean methods such as Principal Component Analysis (PCA) [8], autoencoders, Linear Discriminant Analysis (LDA) [9] or Singular Value Decomposition (SVD) Such newly generated tables are then aggregated with an additional attribute that stores the table's identifier. Based on such data, a single neural network model is built. The proposed system is hierarchical. At the lower level are models that perform transformation and reduction of attributes. On the other hand, at a higher level, a neural network model is available. Classifying a new object also goes through two levels of hierarchy. Preliminary experiments show that the use of this approach is promising. This approach guarantees the protection of the data, as the raw data is not shared. It should also be noted that a single neural network model is ultimately obtained.

Federated Learning is a decentralized machine learning approach where models are trained collaboratively across multiple devices or nodes without centralizing raw data [10]. This collaborative training process aims to preserve data privacy, making it particularly relevant when sensitive information is involved. Therefore, the approach presented in the paper is related to federated learning. In horizontal Federated Learning [11], each node possesses a subset of the feature space but

shares the same label space. Vertical Federated Learning [12], on the other hand, deals with vertically partitioned data, where different nodes hold complementary information about the same set of instances. In the presented approach, the data is partitioned both vertically and horizontally. Federated Learning often involves heterogeneous and non-identically distributed (non-IID) data across nodes [13]. Addressing the challenges of non-IID data distribution is crucial for achieving effective model convergence and performance. Federated Learning is often combined with differential privacy techniques to enhance privacy guarantees. In this paper, we put less emphasis on data protection and more on classification quality. In addition, federated learning usually has an iterative form based on convergence [14]; here, the model is built only once.

The rest of the paper is organized as follows. In Section 2, we present the details of the model. We also describe the method to classify the test object. The last section is a conclusion and some future plans and the challenges.

2. Methods and models

The approach proposed consists of two steps. In the first step, based on each local table a modified table with an equal number of attributes in each table specified as parameter k is designated. In the second stage we train a global neural network model based on the concatenated modified local tables.

The first step transforms each local table by use of a feature extraction method to obtain resulting tables with homogeneous number of attributes. Suppose a set of n decision tables $D_i = (U_i, A_i, d)$, $i = 1, \ldots, n$ from one discipline is available, where U_i is the universe, a set of objects; A_i is a set of conditional attributes; d is a decision attribute. With a predetermined number of features k, a transformation map T_i^k is built to transform each D_i to a matrix of vector of principal component scores M_i^k. The transformation map converts the vector of values of the object on the original conditional attributes into a completely different vector of values with a different dimension equal to k; each $\mathbf{t} \in M_i^k$ is a vector of values $\mathbf{t} = (t_1, \ldots, t_k)$. Transformation map is not shared among local units during the creation of the central model, for this reason the raw data is protected. Moreover, both the information about the number of attributes present in each original table and what those attributes are is not shared. As was mentioned earlier, for each local table these data can be different. Thus, it is rather impossible to reconstruct the original form of local tables due to the very large number of unknown parameters. Any feature

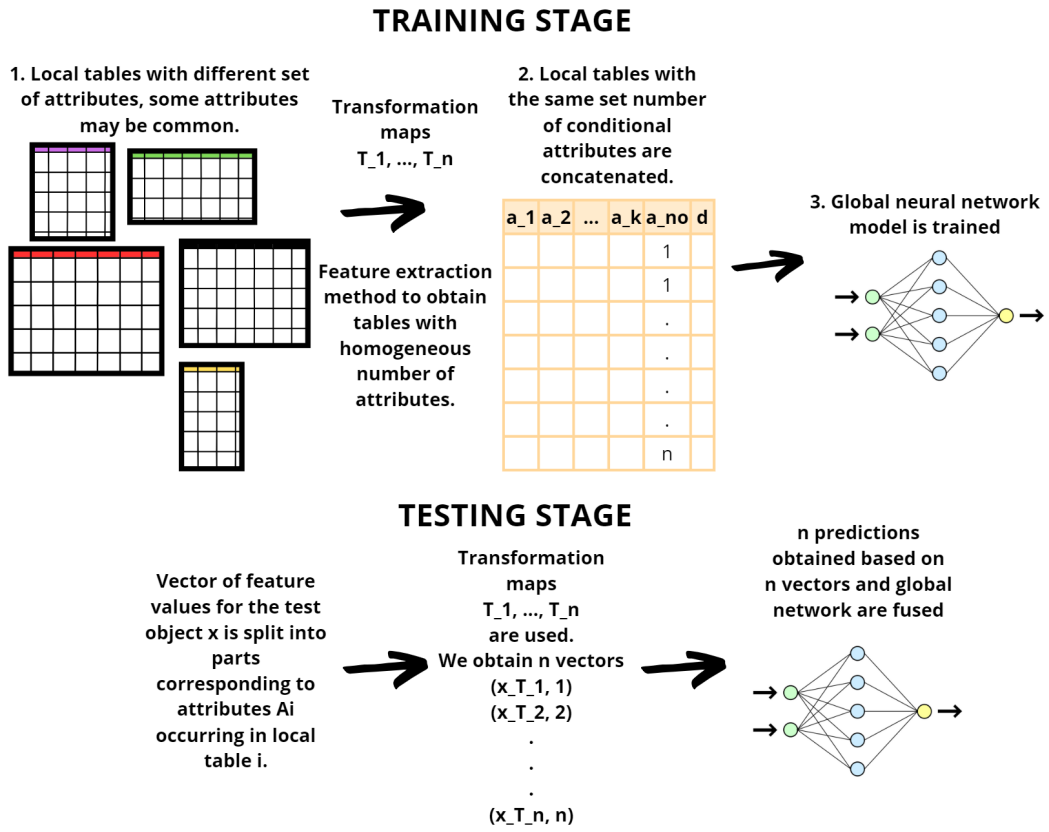
extraction approach such as PCA, LDA or SVD can be used in this step.

In the second stage, all the matrices M_i^k are concatenated (since they have the same number of coordinates) with an additional attribute that stores the table's identifier and label. Concatenated matrix M is used to build a neural network model – second step global model. Different network architectures can be used – from the Multilayer Perceprton (MLP) to Radial Basis Function Networks or Convolutional Neural Networks.

The process of classifying new cases has also two stages. The vector of feature values for the test object is split into parts – corresponding to attributes A_i occurring in local table i. Then the part $\mathbf{x}_i = (a_1(\mathbf{x}), \ldots, a_m(\mathbf{x}))$, where $A_i = \{a_1, \ldots, a_m\}$ are sent to local unit i. The local unit i use the previously determined transformation map T_i^k to obtained $k-$dimensional vector for the test object $\mathbf{x}_{T_i^k}$. The transformed values, along with an additional coefficient denoting the table's identifier are then used as input to the global neural network and prediction is made. In the case where all local tables participate in the classification, we get n predictions for one test object. To obtain the final result, a fusion method such as the sum rule, product rule, the Borda count method, or any kind of voting can used to aggregate the obtained predictions. In this paper, the soft voting used as the fusion method. Figure 1 shows the stages of building the global model and test objects classification process.

In this paper, due to limited space, we only present preliminary experimental results for the approach described above. During experiments, the linear kernel PCA in the first stage and the MLP with a hidden layer in the second stage were used. The tested model's parameters were: the number of principal components generated in the PCA method (from 1 to the minimum number of conditional attributes present in local tables $\min_i |A_i|$ was tested); the number of neurons in the hidden layer ($\{2, 2.5, \cdots, 9.5\}$ times the number of neurons in the input layer was tested). The experimental evaluation was performed on three data sets obtained from the UC Irvine Machine Learning Repository: Vehicle Silhouettes, Landsat Satellite, Dry Bean [15]. Each data available in the repository is stored in a single table. The training data sets are then dispersed into local tables. Different degrees of dispersion are considered in order to check whether the method can cope with significant data dispersion. The versions with $3, 5$, and 7 local tables based on the original training set are considered where all local tables contained only a subset of the original set of conditional attributes. The evaluation of classification performance is conducted on the test set using the classification accuracy (acc), the

Figure 1. Model generation and prediction for test object stages



F-measure (F-m.)

$$\text{F-m.} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

and the balanced accuracy (*bacc*). All these metrics are calculated using functions implemented in Python in the sklearn library and using the weighted average option. All experiments were performed three times, and the results presented in Table 1 are the average value from these three performances. In the table the highest value in terms of accuracy obtained for different tested parameters values are given. As can be seen, the proposed model performed equally well with different degrees of dispersion. In addition, it can be noted that the F-measure is quite high compared to the accuracy. In papers [6, 7] the same data sets are studied using different approaches. For example, in paper [7] the PCA method was not used, and the MLP networks with the same structure were built using artificial objects

Table 1. Results of F-m., *bacc* and *acc* for the proposed approach.

No.	Vehicle			Satellite			Dry Bean		
tables	F-m.	*bacc*	*acc*	F-m.	*bacc*	*acc*	F-m.	*bacc*	*acc*
3	0.677	0.664	0.694	0.419	0.355	0.442	0.903	0.914	0.905
5	0.472	0.464	0.482	0.728	0.701	0.763	0.863	0.873	0.865
7	0.622	0.625	0.644	0.834	0.795	0.848	0.897	0.904	0.898

based on each local table separately. These networks were then aggregated into a single local network. Finally, the global network was post-trained. This approach is much more complex than the one presented in this paper, moreover, additional data are needed to post-train the global network (for details, see paper [7]). It can be concluded that the results presented in this paper are not significantly different from those presented in papers [6, 7]. In addition, the proposed approach is simpler, reduces data dimensionality and ensures data protection and privacy.

3. Conclusions

This paper presents a model for classification based on dispersed data. The system guarantees data protection and reduction of dimensionality by using the PCA for each local table separately. Based on these local models a single global table is created with additional information about the identifier of the table from which the data comes from. A global model is then built. MLP is used in the present study. The paper shows preliminary experimental results which indicates that the proposed model achieves good results. Of course, further experiments are needed to investigate more advanced network structures, as well as the use of different types of neural network models and feature extraction methods. However, preliminary experiments provide promising results.

References

[1] Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. 2016.

[2] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[3] Breiman, L. Random forests. *Machine learning*, 45:5–32, 2001.

[4] Przybyła-Kasperek, M. and Marfo, K. F. Neural network used for the fusion of predictions obtained by the k-nearest neighbors algorithm based on independent data sources. *Entropy*, 23(12):1568, 2021.

[5] Przybyła-Kasperek, M. and Marfo, K. F. Studies on neural networks as a fusion method for dispersed data with noise. In *International Conference on Information Systems Development*, pages 169–186. Springer, 2022.

[6] Marfo, K. F. and Przybyła-Kasperek, M. Radial basis function neural network with a centers training stage for prediction based on dispersed image data. In *International Conference on Computational Science*, pages 89–103. Springer, 2023.

[7] Marfo, K. F. and Przybyła-Kasperek, M. Study on the use of artificially generated objects in the process of training mlp neural networks based on dispersed data. *Entropy*, 25(5):703, 2023.

[8] Ringnér, M. What is principal component analysis? *Nature biotechnology*, 26(3):303–304, 2008.

[9] Hu, P., Peng, D., Sang, Y., and Xiang, Y. Multi-view linear discriminant analysis network. *IEEE Transactions on Image Processing*, 28(11):5352–5365, 2019.

[10] Li, L., Fan, Y., Tse, M., and Lin, K.-Y. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

[11] Huang, W., Li, T., Wang, D., Du, S., Zhang, J., and Huang, T. Fairness and accuracy in horizontal federated learning. *Information Sciences*, 589:170–185, 2022.

[12] Liu, Y., Kang, Y., Zou, T., Pu, Y., He, Y., Ye, X., Ouyang, Y., Zhang, Y.-Q., and Yang, Q. Vertical federated learning: Concepts, advances, and challenges. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[13] Zhu, H., Xu, J., Liu, S., and Jin, Y. Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390, 2021.

[14] Nguyen, H. T., Sehwag, V., Hosseinalipour, S., Brinton, C. G., Chiang, M., and Poor, H. V. Fast-convergent federated learning. *IEEE Journal on Selected Areas in Communications*, 39(1):201–218, 2020.

[15] Dua, D. and Graff, C. Uci machine learning repository [http://archive. ics. uci. edu/ml]. irvine, ca: University of california. *IEEE transactions on pattern analysis and machine intelligence*, 1(1):1–29, 2019.

Publication [P9]

Przybyła-Kasperek M.,Marfo K.F. A multi-layer perceptron neural network for varied conditional attributes in tabular dispersed data *PLoS ONE*, 19:1-56, 2024.
URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0311041.
DOI: 10.1371/journal.pone.0311041.
$\text{MEiN}_{2024} = 100$
Number of citations:

- according to Web of Science: 3
- according to Google Scholar: 3

Contributor	Description of main tasks
Małgorzata Przybyła-Kasperek	- conceptualization and methodology - result analysis - manuscript: original draft preparation - manuscript: review and editing - visualization: creating all figures in manuscript
Kwabena Frimpong Marfo	- investigation - conceptualization and methodology - result analysis - implementation of proposed algorithm - manuscript: review and editing - visualization: creating all figures in manuscript

PLOS ONE

RESEARCH ARTICLE

A multi-layer perceptron neural network for varied conditional attributes in tabular dispersed data

Małgorzata Przybyła-Kasperek ☯ *, **Kwabena Frimpong Marfo** ☯

Institute of Computer Science, University of Silesia in Katowice, Sosnowiec, Poland

☯ These authors contributed equally to this work.
* malgorzata.przybyla-kasperek@us.edu.pl

OPEN ACCESS

Citation: Przybyła-Kasperek M, Marfo KF (2024) A multi-layer perceptron neural network for varied conditional attributes in tabular dispersed data. PLoS ONE 19(12): e0311041. https://doi.org/10.1371/journal.pone.0311041

Editor: Kalapraveen Bagadi, Vellore Institute of Technology - Amaravati Campus: VIT-AP Campus, INDIA

Received: December 10, 2023

Accepted: September 9, 2024

Published: December 2, 2024

Peer Review History: PLOS recognizes the benefits of transparency in the peer review process; therefore, we enable the publication of all of the content of peer review and author responses alongside final, published articles. The editorial history of this article is available here: https://doi.org/10.1371/journal.pone.0311041

Data Availability Statement: The data used in the article is publicly available from the UCI repository. The specific links are as follows: Vehicle

Abstract

The paper introduces a novel approach for constructing a global model utilizing multilayer perceptron (MLP) neural networks and dispersed data sources. These dispersed data are independently gathered in various local tables, each potentially containing different objects and attributes, albeit with some shared elements (objects and attributes). Our approach involves the development of local models based on these local tables imputed with some artificial objects. Subsequently, local models are aggregated using weighted techniques. To complete, the global model is retrained using some global objects. In this study, the proposed method is compared with two existing approaches from the literature—homogeneous and heterogeneous multi-model classifiers. The analysis reveals that the proposed approach consistently outperforms these existing methods across multiple evaluation criteria including classification accuracy, balanced accuracy, $F1-$score, and precision. The results demonstrate that the proposed method significantly outperforms traditional ensemble classifiers and homogeneous ensembles of MLPs. Specifically, the proposed approach achieves an average classification accuracy improvement of 15% and a balanced accuracy enhancement of 12% over the baseline methods mentioned above. Moreover, in practical applications such as healthcare and smart agriculture, the model showcases superior properties by providing a single model that is easier to use and interpret. These improvements underscore the model's robustness and adaptability, making it a valuable tool for diverse real-world applications.

1 Introduction

Machine learning (ML) for dispersed data addresses the challenge of analyzing and utilizing data that is scattered across different sources, formats, and locations. This is increasingly important in the era of big data where data is often inconsistent, heterogeneous, and subject to privacy regulations. Data collected from various sources often lack a uniform structure, in that attributes and objects might differ significantly from one data set to another. In the healthcare sector, data on patients are stored across multiple hospitals and medical facilities, each with its

Silhouettes: https://archive.ics.uci.edu/dataset/149/statlog+vehicle+silhouettes Dry Bean: https://archive.ics.uci.edu/dataset/602/dry+bean+dataset Sensorless Drive Diagnosis: https://archive.ics.uci.edu/dataset/325/dataset+for+sensorless+drive+diagnosis Crowd Sourced: https://archive.ics.uci.edu/dataset/400/crowdsourced+mapping.

Funding: The author(s) received no specific funding for this work.

Competing interests: The authors have declared that no competing interests exist.

own data management system. These data sets might differ in structure, terminology, and format. Additionally, data protection regulations (e.g., HIPAA in the U.S., GDPR in Europe) prevent the sharing of sensitive patient information between institutions without proper safeguards. Suppose multiple hospitals are working together to develop a predictive model for identifying patients who are at high risk of sepsis. Each hospital has its own data set which includes varying attributes such as patient vitals, lab results, and medication histories. The problem is how to collaboratively train a sepsis prediction model leveraging on these diverse data sets. A very important aspect is the ability to make use of inconsistent data that is available in dispersed form, however, due to the arbitrariness of attributes and objects present in local data sets as well as data protection laws that restrict the free flow of data, one has to be meticulous when dealing with dispersed data. ML for dispersed data is crucial in leveraging the full potential of big data across domains where data is fragmented and regulated. It enables organizations to collaboratively develop sophisticated models.

To clarify, certain assumptions have been made that define the area of the problem under consideration. To begin with, there is the assumption that data are available in tabular and dispersed form. Also, data are provided by independent entities that do not want to share data—storing data on a central server with other data from other sources. For this reason, there are a set of local decision tables where objects and conditional attributes of such tables need not satisfy any constraints—they do not have to be equal or same but they may have some shared attributes as well as objects. In the situation considered in the paper, we do not guarantee full confidentiality. We assume that the local entities agree to disclose information about what attributes are stored in the table—the names of these attributes and certain characteristics of the values stored in the table such as mean, median, minimum and maximum of the attributes in the local tables.

Researchers have been seeking for solutions associated with dispersed data in domains such as federated learning and distributed learning. Federated learning puts a greater emphasis on data protection. The general approach here is to distribute an initial model from a central server to all local spaces for the model to be trained locally. Trained parameter values from all local models are then sent to the central server where some aggregations are performed to produce a global model. The global model is then sent back to the local spaces for verification—local units can accept, modify or reject the global model. Such a process is performed iteratively until some acceptable convergence metric is achieved. In [1], a detailed description can be found. For such a global model to be constructed in federated learning, the assumption of an equal set of conditional attributes present in all local tables must be satisfied. Distributed learning on the other hand assumes that all data are available in a centralized form, for example in a single decision table (see [2]). The division into local sets is intentional and aims to improve the quality of the model's classification or its ability to deal with huge data. Often, the process of creating local tables in distributed learning is focused on strengthening the local classifiers by sensitizing them to difficult cases. This approach assumes full access to all data and does not necessarily guarantee any data protection.

The model proposed in this paper is different from the two domains mentioned above. Namely, the proposed model does not impose any assumptions of homogeneity on the form of data present in local spaces whiles guaranteeing a certain level data protection. By sharing data on attribute names and general characteristics of the values stored in the tables, Individual data tuples and individual raw data are protected. Also, the proposed method does not employ an iterative process to reach a consensus but rather a non-iterative algorithm that leads to the construction of a global model.

The main contribution of the paper is to propose a method that generates a global neural network model based on dispersed data. To begin, local neural networks with the same

structure are trained based on local tables where local tables have varied and unrestricted form —no constraints on the set of objects and the set of attributes. In order to generate local networks with the same structure, it is necessary to somehow modify the local tables. The goal is to generate local networks whose input layers considers the full set of conditional attributes present in all local tables. In each of the local tables, values for missing attributes are imputed by using certain characteristics determined based on other local tables containing the missing attributes. In this way, a set of extended local tables are prepared and used to train local MLP networks. This study uses the MLP networks as this is the initial take on the proposed approach, thus, it is appropriate to start with the standard neural network suitable for classification. Other types of networks such as the radial basis function neural networks as well as autoenconders are planned to be used in future work. In the next step, the local neural networks are aggregated. In this study, two approaches of aggregating networks are considered—average and sum of the weights from the local models. Finally, the aggregated model is retrained with a sample of data that is shared and defined for the full set of conditional attributes. In this way the final global model is constructed.

In this paper, the proposed model is investigated in terms of many variants. The following features are tested:

- the method to substitute values of missing attributes in local tables (different number of artificial objects generated based on one original object are tested),
- the number of hidden layers in MLP networks (k-hidden layer networks are tested, $k \in \{1, 2\}$),
- the number of neurons in the hidden layer/s (different values are tested),
- the method of aggregating local neural networks (sum and average are tested).

The proposed method is compared with other methods from the literature to establish its quality. Two approaches are adopted as baseline methods. To begin, the ensembles of classifiers proposed in the paper [3]. It comprises creating three base classifiers: k-nearest neighbors, decision tree and naive bayes classifier (KNN, DT, NB) based on each local table. The final decision is made by voting. The second approach is a homogeneous ensemble of classifiers and consists of generating MLP networks based on each local table separately and then generating the final decision by voting. It is shown in this paper that the proposed model produces much better results than both baseline models. These differences are also confirmed with statistical tests.

The paper is organized as follows. In the second section, an overview of the literature is included. The third section presents the newly proposed adaptive approach. Here, a formal definition of dispersed data and description of steps of the process of building a global model based on dispersed data is included. The fourth section gives the experimental protocol and description of the data used. The fifth describes obtained results. Comparative analysis are also carried out in this section. Finally, a summary is presented in the conclusion section.

2 Related work

Artificial intelligence (AI) has transformed various sectors by integrating human-like abilities such as learning, reasoning, and perception into software systems. This advancement has enabled computers to execute tasks traditionally performed by humans. Fueled by enhancements in computational capacity, the availability of extensive data sets and the creation of state-of-the-art AI algorithms, AI applications have become widespread. Noteworthy examples

include finger vein recognition [4], diabetic retinopathy detection [5], RNA Engineering [6], cancer detection [7], biomathematical challenges [8], and smart agriculture [9].

Ensemble learning deals with distributed data which is similar to the issues in this paper. It is a very popular technique in machine learning that is employed to boost the predictive performance of learning algorithms. The underling reasoning of using this approach is to tackle problems involving data sets that are too large to handle at once [10] or in situations where having access to a very small data set, at which data sampling is necessary to obtain reasonable results [11]. Another rationale for using this approach may be to cope with the issue of identifying the right model for the considered problem [12]. To expound, rather than risking selecting the wrong model, one can use a heterogeneous approach of ensemble learning. This approach also works well for problems whose solution space is quite large, thus, faces the risk of getting stuck in local minima/maxima [13]. Many different approaches involving the use of neural networks to address the above mentioned problems have been proposed. Such solutions are proposed in areas such as the business field [14], malware detection [15] and audio classification [16], however, all these approaches assume free access to data and a necessary condition that all data is stored in a centralized form rather than a dispersed one.

Federated learning is another approach within distributed machine learning [1]. Different from classifier ensembles, it puts the greatest emphasis on data segregation and protection [17]. Here, the assumption is that data are available in separate sets that must not be centralized. The idea is to build local models separately and generate a global model in a central space by iteratively aggregating the local models. Neural networks are well applicable here as it is relatively simple to aggregate these models while maintaining high quality [18, 19]. There are types of federated learning: horizontal, vertical and hybrid federated learning. The latter approach is the closest to the approach proposed in this paper, however, unlike the proposed approach, hybrid federated learning requires that different parties share the data identity information which is a threat to the privacy of local clients [20]. Unfortunately, for the considered data sets, it is impossible to apply this approach due to the hybrid nature of the partitioning—regarding both objects and attributes—and the inability to obtain identity information about objects between dispersed data sets. Many different models are proposed in federated learning with various aggregation methods, network types and applications being considered in the literature [21–23].

Another approach to the problem of classification based on dispersed data is to build a separate model that aggregates prediction vectors generated by independent local models. Data privacy is also preserved here as only prediction vectors are consolidated. The form of the data can be completely arbitrary in this approach but here, a global model is not generated and the algorithm is non-iterative. Instead, it generates a separate model that only aggregates the prediction results obtained by local models. The local models can be of a completely different type than the aggregation model. In the literature, one can find papers that use neural networks, decision trees or other models as the aggregation model [24–28]. Statistical as well as dynamic approaches to this issue are also proposed which also consider conflicts or compatibility of local classifiers [29–31]. However, in the present study, the approach considered is different as the goal is to determine a global model based on dispersed data.

MLPs have been key in developing neural networks and machine learning. Although more complex models like Convolutional Neural Networks (CNNs) and Transformers have emerged, recent improvements have renewed the importance and usefulness of MLPs particularly where simplicity and efficiency are needed. Techniques such as Adam and RMSProp [32] have enhanced MLP training by dynamically adjusting learning rates, leading to faster convergence and improved generalization. Incorporating residual connections within MLPs akin to ResNet architectures [33] has mitigated the problem of gradient vanishing, enabling the

training of deeper MLP models. MLPs traditionally require large amounts of labeled data to perform effectively. Techniques such as data augmentation and transfer learning are being adapted to address this limitation [34]. Some of the techniques mentioned above (e.g Adam optimizer [35]) are used in this paper for MLP. But, to the best of our knowledge, MLP networks have never been used in the way that is proposed in this paper—for dispersed data with different sets of attributes using augmentation of missing attribute values.

3 Basic concepts and proposed global model

In this section, we present preliminary designations as well as a detailed discussion on the proposed method for generating a global MLP network model based on dispersed data.

3.1 Dispersed data

A necessary assumption made is that data are available in a dispersed form—separate independent predefined data sets which are free of any constraints. In real applications, independent units collect data in tabular form. In tables, both sets of conditional attributes and sets of objects do not necessarily have to be disjoint as they may share common elements.

Also, there is an assumption that a set of decision tables is given. The tables are collected independently by separate units. A set of decision tables—local tables $D_i = (U_i, A_i, d)i \in \{1, \ldots, n\}$ from one discipline is available, where U_i is the universe, a set of objects; A_i is a set of conditional attributes and d is a decision attribute. Decision tables are collected independently so both sets of objects and sets of attributes can have any form. They can have common elements between tables, but not necessarily. The only condition that must be satisfied by all local tables is the collection of data from one discipline. Formally, this is satisfied by the assumption that the same decision attribute is present in all tables.

Since different sets of attributes appear in local tables, the construction of a MLP local model based on each of the tables separately would create a set of networks with completely different structures. This is because the input layer in each neural network would be different since the feature vectors are not the same across the local tables, thus, making it impossible to aggregate local MLP networks into a single global model.

The approach proposed in this paper is completely different from previous studies as it has not been proposed in the literature until now. The steps of the approach are listed below.

1. Determine a uniform MLP network structure for a set of local tables—dispersed data;
2. Train a MLP network based on each local table separately.
3. Aggregate MLP networks into a single model—a global MLP network;
4. Post-train the global MLP network with a sample of global data.

Fig 1 shows the general steps of building the global MLP network model from dispersed data. In the first step, there is dispersed data—local tables with different sets of conditional attributes and different sets of objects. In order to build local neural networks with the same structure (the input layer requires the most attention here), the training data in each local space is imputed so as to have the same set of attributes. This step is carried out with the help of certain characteristics calculated from local tables. It is important to emphasize that the raw data is not shared at any model construction stage. In the next step, local MLP networks are trained, after which they are aggregated to construct a global network. The final step is to re-train the global network. In the study, this is done using a validation set.

All the steps are discussed in detail in the subsequent subsections.

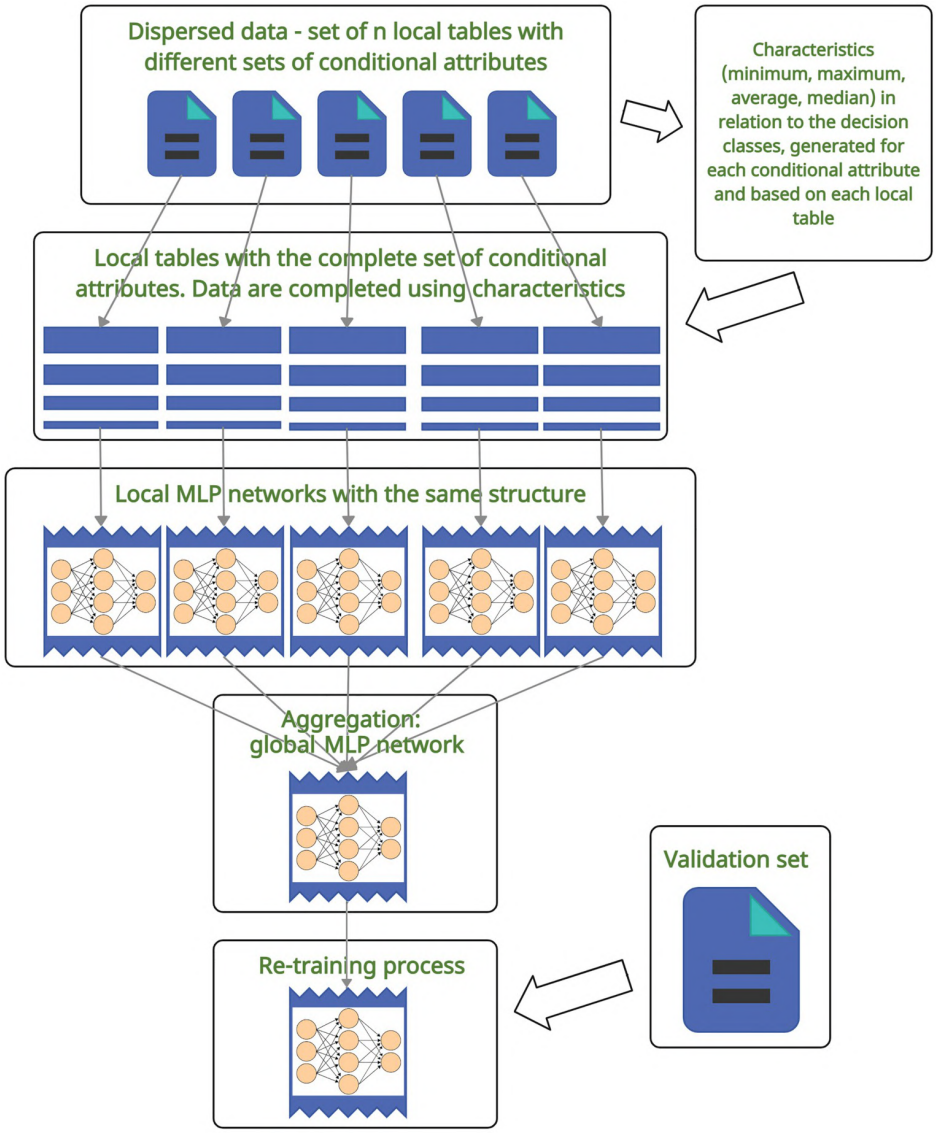


Fig 1. Stages of building the global MLP network model based on dispersed data.

https://doi.org/10.1371/journal.pone.0311041.g001

3.2 Determine an uniform MLP network structure for a set of local tables—local models

Since the dispersed data need not satisfy any constraints, the key in determining the structure of the MLP network is in the number of neurons in the input layer. The output layer poses no problem since all local tables share the same decision attribute. The number of hidden layers as well as the number of neurons in the hidden layers are optimized experimentally. Thus, the most important challenge is to determine a common input layer. In this first study on the approach, it is proposed to unify the input layer by using all conditional attributes from local tables. So the input vector will have the dimension determined by the number of elements in

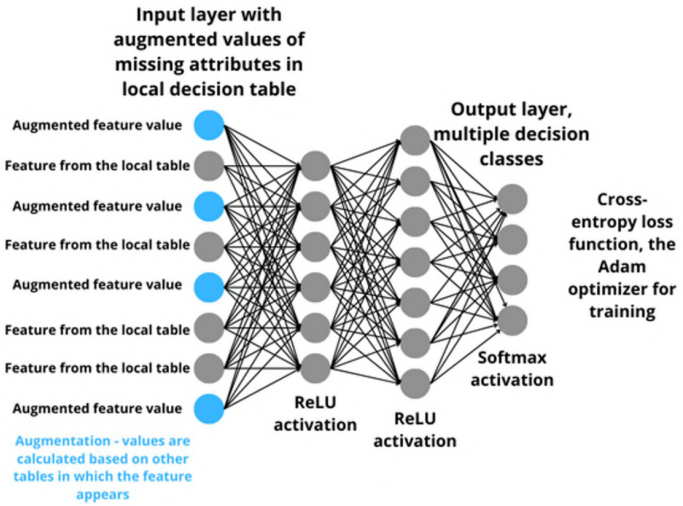


Fig 2. Schematic diagram of MLP network structure for one local decision table.

https://doi.org/10.1371/journal.pone.0311041.g002

the sum of conditional attributes present in the local tables

$$\text{Number of neurons in the input layer} = \text{card}\left\{\bigcup_{i=1}^{n} A_i\right\},$$

where card{X} is the number of objects in the set X. Such a sum is not a simple concatenation of attributes. We operate on sets, and we recognize attributes by their names. So the sum of the sets skips multi-duplicates—in case when one attribute appears in several tables it only appears once in the sum. It should also be noted that such a sum does not mean summing tuples from a table, but only determining the set of names of all attributes appearing in local tables.

Here a problem arises because local tables contain objects for which values are known only on a certain subset of the set $\bigcup_{i=1}^{n} A_i$. The question arises on how to train the local MLP networks with the input layer defined as above based on a local table with such objects. Fig 2 shows the overall configuration of the MLP network—local model used for each local decision table. In each of the local tables, a certain number of attributes (features) are included but not all of them. In order to make the network structure common for all local tables, the completion of missing values for a given local table is made. Of course, in each local table other missing values may occur. Completion of missing values is carried out by calculating values from local tables in which the attribute occurs. Local models are neural networks trained specifically on artificially created objects. That is, those that have completed values on attributes that are not present in the actual given local table. So only these artificial objects are used to train the neural network, the original objects are not used. The training process for these models involves a standard neural network built using the Keras library in Python, employing backward propagation over multiple epochs and steps within each epoch. In the next section, an explanation on how this problem is solved is given.

3.3 Training a MLP network based on each local table separately

In this section, the explanation on how to train a local MLP network based on a local table is given. Let us assume that a local table $D_j = (U_j, A_j, d)$ is given, based on which a local MLP network is to be trained with an input layer containing card$\{\bigcup_{i=1}^{n} A_i\}$ neurons. For an object $\bar{x} \in$

U_j from the local table D_j, values for attributes from the set A_j are specified, which means for each $a \in A_j$ value $a(\bar{x})$ is given. Thus, in order to provide an input vector to the MLP network, the values on the other attributes from the set

$$\bigcup_{i=1,i\neq j}^{n} A_i$$

must be determined. Let us assume that attribute b belongs to the set $\bigcup_{i=1,i\neq j}^{n} A_i$ and for this attribute one has to determine the value to be completed for the object $\bar{x}$. In the proposed approach, this value is determined based on certain statistical measures: minimum, maximum, median and average calculated for values of attribute b occurring in other local tables in the dispersed data. In addition, the decision class of the object $\bar{x}$ is also taken into account. These measures were chosen as the most popular, frequently used in numerous calculations and characterize both the central tendency and the entire range of variation in the value of a given attribute. The paper is the first study of the approach using artificial objects. In future work, other statistical measures will be analyzed. It is planned to use quartiles and the average value offset by the standard deviation.

More strictly, let us assume that the object $\bar{x}$ has a decision value v, $d(\bar{x}) = v$, $v \in V^d$, where V^d is the set of values of the decision attribute d. For each of the decision tables to which the attribute b is present, the minimum, maximum, median and average are calculated for the values of the attribute b based on the objects in the decision class v. For each decision table D_i for which $b \in A_i$ the following values are computed:

$$MIN_{i,v}^{b} = min_{x\in U_i, d(x)=v}\ b(x), \quad MAX_{i,v}^{b} = max_{x\in U_i, d(x)=v}\ b(x),$$

$$AVG_{i,v}^{b} = avg_{x\in U_i, d(x)=v}\ b(x), \quad MED_{i,v}^{b} = median_{x\in U_i, d(x)=v}\ b(x)$$

In this way, values designated separately for each local table containing the attribute b are obtained. To determine the final value which is completed in the object $\bar{x}$ and given to the input of the neural network, one of the statistical measures (minimum, maximum, mean or median) is applied on the local values determined in the previous step. Thus, one of the four measures for determining local values based on local tables and one of the four measures for determining the aggregate value. In all, there are 16 possible combinations from which one is chosen at random as the value of b. Suppose that for calculating local values the median is drawn, and for aggregate value, the minimum is drawn, then the value on attribute b is determined as follows

$$b(\bar{x}) = min_{D_i: b\in A_i} MED_{i,v}^{b}.$$

This method is repeated for each of the missing attributes $\bigcup_{i=1,i\neq j}^{n} A_i$ for object $\bar{x}$.

In the generalized version of the above method, instead of one object, $k (k < = 16)$ objects are generated by selecting k distinct values from the 16 possible values as the value of b in each of the k objects generated from the original object $\bar{x}$. Thus, based on object $\bar{x}$, k new objects would be generated with all values on conditional attributes $\bigcup_{i=1}^{n} A_i$. This approach is also tested and the results are presented in the experimental part of the paper. Algorithm 1 presents the pseudo-code of the generalized version (in the basic version, it is enough to put $k = 1$), which implements this part of the model.

Algorithm 1 Pseudo-code of algorithm generating objects from one local table used for training the local MPL network

Input: One local decision table $D_j = (U_j, A_j, d)$ for which we determine the training set for the MLP network; measures minimum, maximum, median and average $MIN^b_{i,v}$, $MAX^b_{i,v}$, $MED^b_{i,v}$ and $AVG^b_{i,v}$ computed for each decision value $v \in V^d$ and attribute $b \in A_i$ based on the values stored in the table D_i for each $i \in \{1, \ldots, n\}$; $A = \bigcup_{i=1}^{n} A_i$ a set of conditional attributes from all local tables; k parameter value that determines how many objects are generated based on one object from table D_i.
Output: A data set used to train the MLP neural network, $\overline{D_j} = (\overline{U_j}, A, d)$.

```
foreach x ∈ U_j
  for m = 1 to k do
  create an object x̄ from D̄_j by assigning values on the set A_j the same
as the object x has
    foreach attribute in the set b ∈ A\A_j
      choose a pair (choice1, choice2) from the set
      {MIN, MAX, AVG, MED} × {MIN, MAX, AVG, MED}
      b(x̄) = choice2_{i:b∈A_i}(choice1^b_{i,d(x)})
    end foreach
  end foreach
end foreach
```

The computational complexity of the above method is linearly dependent on the number of objects in the local table D_j, value of parameter k, the number of conditional attributes card$\{A\}$ and the number of local tables in the dispersed data n. More precisely, the complexity resulting from the loop is $O(\text{card}\{U_j\} \cdot k \cdot \text{card}\{A \backslash A_j\} \cdot \text{card}\{D_i, i = 1, \ldots, n\})$. In the worst case, one can assume that there is only one conditional attribute in the table D_j, and for all other attributes, values have to be computed and the missing attributes are present in all other local tables except D_j. Then the complexity is $O(\text{card}\{U_j\} \cdot k \cdot (\text{card}\{A\} - 1) \cdot (n - 1))$. The linear complexity of the algorithm proves that it can be used even for large dispersed data.

The data prepared in the above way in the next step is used for training MLP neural networks. As mentioned earlier, the input layer is defined by a set of conditional attributes from all local tables. The number of neurons in the output layer is equal to the number of decision classes. Each of the neurons determines the probability with which the test object belong to a given decision class. In the experimental part, one or two hidden layers are considered. The number of neurons in the hidden layer is determined in proportion to the number of neurons in the input layer. Different proportions are checked from 0.25 to 5 times the number of neurons from the input layer. In the case of two hidden layers, all combinations of the number of neurons in the hidden layers are checked such that: the first layer had the number of neurons from the set $\{0.25 \times I, 0.5 \times I, 0.75 \times I, 1 \times I, 1.5 \times I, 1.75 \times I, 2 \times I, 2.5 \times I, 2.75 \times I, 3 \times I, 3.5 \times I, 3.75 \times I, 4 \times I, 4.5 \times I, 4.75 \times I, 5 \times I\}$, and the second layer had the number of neurons from the set $\{1 \times I, 2 \times I, 3 \times I, 4 \times I, 5 \times I\}$ where I is the number of neurons in the input layer. For the hidden layer, the ReLU (Rectified Linear Unit) activation function is used, as it is the most popular activation function and gives very good results [36]. For the output layer, the softmax activation function is used, which is recommended when one deals with a multi-class problem [37]. In this paper, data sets containing from four to nineteen decision classes are analyzed. The neural network is trained by using the back-propagation method. A gradient descent method, with an adaptive step size is used in the back-propagation method. It is known that the softmax layer give good results with the Adam optimizer [35]. The Adam optimizer proposed in [38] and is one of the most popular adaptive step size methods. From [39], the categorical cross-entropy loss gives best results with softmax layer. That is why the Adam optimizer and the categorical cross-entropy loss function are used in the study.

The implementation of the MLP neural network from Keras library in Python is used. The algorithm that defines a neural network with one or two hidden layer with the rectified linear

unit (ReLU) activation function and the number of neurons in the first hidden layer dependent on the parameter. Softmax activation function is used in the output layer. In the compilation, the categorical cross-entropy loss function, the Adam optimizer and the accuracy as the learning rate are used. For two hidden layers approach the second hidden layer with the ReLU activation function and the number of neurons dependent on the parameter is used. In the way described above, a set of local MLP networks are obtained. The number of networks is equal to the number of local decision tables. All networks have the same structure and this is a very important property necessary for the next step.

3.4 Aggregation of MLP networks into a single model—a global MLP network

The result of the previous stage is a set of local MLP networks which are trained and all have the same structure. Aggregation of such networks into a single global MLP model is relatively simple. The global network has exactly the same structure as each of the local networks i.e. the same number of layers and the same number of neurons in each layer. However, during aggregation, each local model may have a different impact on the construction of the global model. This influence is proportional to the quality of each local model's classification on the training set. The method used is inspired by the second weighting system used in the AdaBoost algorithm [40].

For each local model, a classification error is estimated based on its training set (artificial objects generated using Algorithm 1). Let us denote by e_i the classification error determined for the i–th local model $i \in \{1, \ldots, n\}$. Since local models are built based on a piece of data, their accuracy can be very different. It may sometimes happen that their classification error is above 0.5. In order not to eliminate such local models from the aggregation stage as they may contain important information on specific attributes that may have a positive impact in the global model, the min-max normalization is applied to the interval [0, 0.5] of all errors e_i, $i \in \{1, \ldots, n\}$. After, the weights ω_i for each local neural network $i \in \{1, \ldots, n\}$ is adjusted according to the formula:

$$\omega_i = ln\left(\frac{1 - e_i}{e_i}\right) \tag{1}$$

The weights of global model are determined by one of two approaches: in the first approach, the weights for the global network are determined by the weighted average of the corresponding weights (assigned to edges connecting exactly the same neurons) present in local MLP networks with weights ω_i, $i \in \{1, \ldots, n\}$. The second approach is to determine the weight for the global network as the sum of the corresponding weights from the local networks with weights ω_i, $i \in \{1, \ldots, n\}$. The two approaches are studied separately in the experimental part of the paper.

Fig 3 illustrates the process of aggregating local models into a global model. Since all local models share the same structure, this aggregation is relatively straightforward. Each connection between neurons in the global model corresponds to the connections in the local models. The critical aspect of this process is the determination of weights, which are based on the classification performance of local models on their respective sets of artificial objects (training sets). The weights assigned to each local model are crucial as they influence the global model's configuration. Local models that perform poorly in classification (possibly due to a higher number of missing attributes and thus less connection with reality, more values are artificial) are given smaller weight in shaping the global model. However, they are not entirely excluded from the aggregation. It will still contribute to the overall classification performance of the

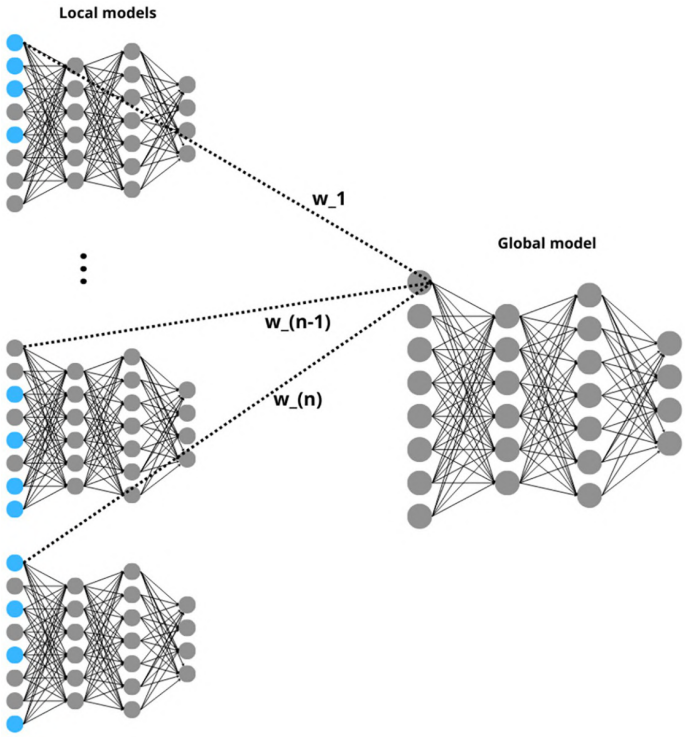


Fig 3. Schematic diagram of global model.

https://doi.org/10.1371/journal.pone.0311041.g003

global model. This ensures that the global model benefits from the specialized capabilities of each local model, enhancing its overall classification quality.

The implementation of the global MLP network is done in Python. First the network's structure is defined—the number of layers and neurons in the layers are the same as in local networks. Then the weights are not trained but assigned based on average or sum with consideration of the weights for the local networks of corresponding connections in local networks.

3.5 Re-training the global MLP network with a sample of data

The retraining process with global objects is a step that enhances the proposed model's accuracy, generalization, and robustness. By carefully integrating and fine-tuning the local models, the global model achieves superior performance, making it a valuable tool for various real-world applications. This process integrates local models into a cohesive global model, enhancing overall accuracy and generalization. The global MLP network is re-trained using the validation set. This step involves adjusting the weights and biases of the aggregated model to fine-tune it for better performance. The retraining process ensures that the global model leverages the strengths of the local models while mitigating their individual weaknesses. A validation set, which is a subset of the training data, is used for the retraining process. The size of this validation set is smaller than the local models' training sets but is crucial for capturing the model's generality. The validation set helps in fine-tuning the global model to prevent overfitting and ensure it generalizes well to unseen data. What is important is that the objects in such a validation set must contain a global description of the objects, i.e. include attributes/characteristics present in all local tables. The integration of local models through retraining results in a

significant boost in classification accuracy. The model benefits from the collective knowledge of all local datasets, leading to more accurate predictions.

The last stage is to re-train the global network. The training objects needed in this step should have values on the set of all conditional attributes $\bigcup_{i=1}^{n} A_i$. In the paper, this is implemented by using a validation set. Such a validation set is much smaller than the training sets for local models and will have less influence on the final form of the global neural network. However, without the use of this last step, the obtained quality of classification is unsatisfactory and we miss to capture a generality of the model. For the approach of generating one artificial object, the size of a validation set is about 21% of the size of the training set for local model. In the case of generating three artificial objects, the size of a validation set is about 7% of the size of the training set for local model. In future works, it is planned to test the active learning approach [41, 42] instead. In active learning, the assumption is that the model builds its own training data or changes the original training data.

After the completion of this step, the final form of the global model is obtained and is evaluated by using an independent test data set.

It should be noted that the model avoids overfitting through a series of carefully planned steps. The final stage of training involved re-training the global MLP network with a validation set. This validation set is smaller than the training sets for local models, but crucial in capturing the generality of the model. For generating one artificial object, the validation set os about 21% of the local model's training set size. For three artificial objects, it is about 7%. This step is essential to prevent overfitting and ensure the model could generalize well to new, unseen data. Also during the selection of optimal parameters, we aimed for the best classification accuracy with the lowest possible model complexity, involving the fewest layers and neurons. This focus on simplicity helped in reducing the risk of overfitting.

4 Experimental setup

In order to assess the efficiency of the suggested model, the methodology of the experiment is analyzed within this section. The simulation platform, parameter allocation, and criteria for measuring performance are all elaborated upon. The scheme for describing the experimental methodology, which is widely used in the literature as shown in the work of [43, 44] is used below.

4.1 Simulation platform

All simulation is conducted using an open-source software Jupyter Notebook 6.5.4 and Anaconda 2023.09-0 (Sep 29, 2023) Installer Python Version: 3.11.5. The implementation of the proposed model is made using the Keras library in Python. The simulations were run on a computer with an Intel(R) Xeon(R) W-2235 CPU @ 3.80GHz 3.79 GHz processor and 32.0 GB RAM to avoid any bias in the analysis of the results, it is crucial that the study is conducted using the same compiler, on the same computing hardware, and with the same processing capabilities.

4.2 Data set

The experimental study uses data sets available in the UC Irvine Machine Learning Repository: Vehicle Silhouettes [45], Dry Bean [46], Sensorless Drive Diagnosis [47] and Crowd Sourced [48]. The characteristics of the data sets are given in Table 1.

Each of the data sets are originally available in non-dispersed form—each data in a single decision table. The training set are dispersed where different degrees of dispersion are considered. Each single data set is converted into five different dispersed versions: 3, 5, 7, 9 and 11

Table 1. Basic characteristics of data sets.

Data set	No of examples in training set	No of examples in test set	No of attributes	No of decision classes
Vehicle Silhouettes	592	254	18	4
Dry Bean	9527	4084	17	7
Sensorless Drive Diagnosis	38509	20000	49	11
Crowd Sourced	10546	300	29	6

https://doi.org/10.1371/journal.pone.0311041.t001

local tables respectively. During the construction of the local tables, a subset of attributes in each local table is considered. The number of attributes is significantly reduced in local tables as compared to the original table with some attributes repeating among some tables to satisfy. This is done to make provision for the possibility that some local tables may share common attributes. The full set of objects is stored in each local table but without their identifiers. More precisely, the number of local tables is first determined (e,g dispersed version with 5 local tables). Then the number of original set of conditional attributes is divided evenly among the local tables (so that each local table had more or less the same number of attributes). In addition, it is assumed that there are common attributes between the selected local tables, e.g. between table one and two we have two common attributes, between table two and three we have one common attribute, and so on. With the initial assumptions made, attributes are then randomly distributed between local tables. Once we have established sets of attributes in local tables then entire columns from the original tables are rewritten into local tables. In this way, we have the same sets of objects in all local tables.

The Sensorless data set is balanced with each decision class containing 5319 objects. The Vehicle, Dry Bean and Crowd Sourced data sets are imbalanced (Fig 4). The data are balanced but it is worth emphasizing that this process is carried out after the dispersion (to keep the approach as consistent with the real situation as possible). The Synthetic Minority Over-sampling Technique (SMOTE) method is used [49] for each local decision table separately. The implementation of this algorithm available in WEKA [50] software is used. The data considered are multiclass labeled so in each decision table and for each decision class except the most dominant one, the SMOTE method is used. As a result, all decision classes have the same number of objects after balancing. Finally, for each of the three original data sets, 5 dispersed versions of imbalanced data and 5 dispersed versions of balanced data are obtained. Thus, a total of 35 dispersed data are considered in the experimental part.

4.3 Parameter assignments

The proposed model comprises of three phases: structure phase, training phase, and testing phase. The structure phase involves determining the structure of local and global MLP models. The input layer of the model is strictly dependent on the data set—the number of neurons is

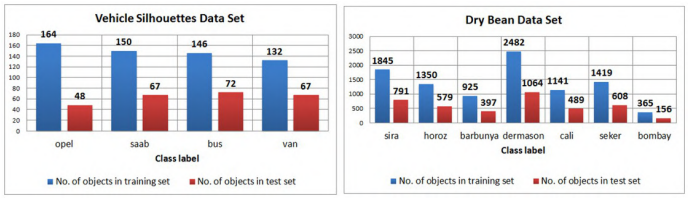


Fig 4. Imbalance of data—cardinality of decision classes in training and test sets.

https://doi.org/10.1371/journal.pone.0311041.g004

equal to the number of attributes present in all local tables. The same is true of the output layer —the number of neurons is equal to the number of decision classes. However, as for the other parameters of the network, it is variable and determined experimentally. Also, the method of determining the value of missing attributes, as well as the number of artificial objects used is variable. Another parameter of the model is the method of aggregation of local networks. Different parameter values are tested. We conducted a comprehensive grid search to explore various configurations of the hyperparameters. This involved testing different combinations of the number of hidden layers, the number of neurons in each layer, methods for aggregating local neural networks, and strategies for handling missing attributes. By systematically varying these parameters, they are able to identify the optimal configuration that achieved the best performance. The optimal number of hidden layers and neurons in each layer is determined through the grid search. The authors tested various configurations, ranging from shallow networks with fewer layers to deeper networks with more layers and neurons. The chosen configuration provided the best trade-off between model complexity and classification accuracy. Different methods for aggregating the local neural networks are evaluated. The authors considered techniques such as averaging the weights of the local models and summing the weights. By employing a systematic and thorough approach to hyperparameter optimization, the authors ensured that their model is both robust and efficient. The experiments are carried out according to the following scheme:

- Different approaches to substitute values of missing attributes in local tables are studied—one or three artificial objects are generated based on one original object in local table.
- Different approaches to aggregating local neural networks are studied—using average of weights or sum of weights.
- Different numbers of hidden layers in the local and global networks are studied—one or two hidden layers.
- Different numbers of neurons in hidden layers are studied. The number is determined in proportion to the number of neurons in the input layer. The following values are tested: for the first hidden layer $\{0.25, 0.5, 0.75, 1, 1.5, 1.75, 2, 2.5, 2.75, 3, 3.5, 3.75, 4, 4.5, 4.75, 5\} \times$ the number of neurons in the input layer; for the second hidden layer $\{1, 2, 3, 4, 5\} \times$ the number of neurons in the input layer.

So in total, 384 different experiments for the proposed method are conducted—384 different settings of the approaches analyzed ($2 \cdot 2 \cdot 16 + 2 \cdot 2 \cdot 16 \cdot 5$). In the tables in Section 5, we show both the results from each parameters settings, as well as the optimal parameter values. The optimal parameters are chosen as those provide the best classification accuracy with the lowest possible model complexity—the lowest number of layers and the lowest number of neurons used in the model.

4.4 Formulations of the performance metrics

The quality of classification are evaluated based on the test set. A classification accuracy measure (*acc*) is used for this purpose. That is, a fraction of the total number of objects in the test set that are classified correctly. As is mentioned in the previous section, in the final step the aggregated model is re-trained. To do this, the use a validation set containing objects that have values on all conditional attributes present in the dispersed data is used—attributes occurring in all local tables. The validation set is obtained by dividing the original test set randomly but in a stratified manner into two equal parts. First, one part is used as the validation set (for re-training process) and the second part is used to assess the quality of classification. Then the

roles reverses as the second part acts as the validation set. Finally, both results are averaged. Each experiment is repeated three times; in the following section, all results given are the average of these three runs.

4.5 Reproducibility of the proposed model

The structure phase aims to prepare the local MLP neural networks with a consistent architecture. This involves identifying common attributes among the local tables and addressing any missing attributes. The next step involves supplementing the local tables with additional objects, assigning values to the missing attributes. Following this, the training phase optimizes the network weights. During the testing phase, the model's accuracy is validated on test objects with values defined for all attributes. To assess the effectiveness of the proposed model, a comparative analysis is performed against baseline training methods. Given the inherent randomness in the structure phase—where missing values are filled and new objects are created—and the training phase—where initial network weights are set randomly—the experiments are repeated three times and the results are averaged. The simulations are configured as follows to facilitate reproducibility. Benchmark data that is publicly available are used and the process of splitting (performed only once) into local tables is done as described as in Section 4.2. The local tables are then augmented with additional values on missing attributes. One setting of all parameters is selected. The experiments are performed three times and results are averaged. Such a procedure is repeated for all other parameter settings using the same predefined local tables.

4.6 Baseline methods

In the literature, there are no models dedicated for dispersed data to generate a single model based on local tables with different attributes (but some of them are common). Therefore, for comparison, an intermediate approach is used, which, although does not generate a global model and does not resolve differences between attributes, it generates local models and performs global classification by voting. In the paper, two approaches for building a local model are used.

- The first approach is an ensemble of homogeneous classifiers, where the base classifiers are MLP networks. However, it should be noted here that each network generated for local tables has a different structure as the input layer is different. Which means that no unification of the input layer is done by filling in the values of missing attributes. For a single local table, an MLP network is created, which in the input layer has neurons corresponding to attributes occurring exactly in that local table. In order to maintain transparency and integrity numbers of neurons in the hidden layer are tested as for the proposed method. The number of neurons in the output layer is the same in all local models as it is equal to the number of decision classes. The final decision of the ensemble is made by soft voting.
- The second approach is the method proposed in [3]. This ensemble of classifiers method consists of creating three base classifiers: k−nearest neighbors, decision tree and Naive Bayes classifier (KNN, DT, NB) based on each local table. The parameter $k = 3$ and the Gini index as a splitting criterion when building decision trees are used. Thus, three classifiers are defined for each local table. The final decision of the ensemble is also made by soft voting.

Both approaches are implemented in the Python programming language using implementations available in the sklearn library.

Table 2. Results of classification accuracy *acc* for the proposed approach: One hidden layer, the method to substitute values of missing attributes in local tables—One artificial objects generated based on one original object, MLP networks aggregation using average of weights and various number of neurons in the hidden layer (1AO-1HL-AVG). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	No. of neurons in hidden layer															
		0.25 × I	0.5 × I	0.75 × I	1 × I	1.5 × I	1.75 × I	2 × I	2.5 × I	2.75 × I	3 × I	3.5 × I	3.75 × I	4 × I	4.5 × I	4.75 × I	5 × I
Vehicle imbalanced	3	0.283	0.612	0.659	0.635	0.644	0.634	0.677	0.675	0.672	0.651	0.668	0.671	**0.693**	0.686	0.671	0.688
	5	0.51	0.665	0.629	0.663	0.678	0.639	0.677	0.681	0.698	0.685	0.673	0.689	0.681	**0.703**	0.681	0.665
	7	0.283	0.682	0.639	0.672	0.661	0.688	0.655	0.686	0.686	0.677	0.678	0.676	0.69	**0.69**	0.661	0.669
	9	0.283	0.623	0.656	0.542	0.648	0.646	0.652	0.655	0.669	0.654	**0.677**	0.656	0.667	0.669	0.648	0.655
	11	0.283	0.487	0.617	0.66	0.643	0.647	0.66	0.623	0.676	0.664	0.652	0.657	0.657	0.669	0.648	**0.696**
Vehicle blanced	3	0.584	0.665	0.642	0.682	0.688	0.711	0.703	0.713	0.694	0.714	0.707	0.723	0.718	0.724	0.69	**0.727**
	5	0.283	0.673	0.682	0.65	0.66	0.657	0.69	0.677	0.664	0.675	0.664	0.688	0.68	**0.703**	0.682	0.693
	7	0.283	0.697	0.688	0.692	0.705	0.701	0.714	0.699	0.709	0.717	0.715	0.72	0.698	0.714	0.714	**0.722**
	9	0.283	0.608	0.689	0.669	0.686	0.711	0.698	0.709	0.697	0.688	0.713	0.724	0.705	0.682	0.713	**0.73**
	11	0.283	0.6	0.663	0.654	0.665	0.675	0.668	0.665	0.682	0.652	0.686	0.668	0.684	0.655	0.669	**0.688**
Dry Bean imbalanced	3	0.841	0.888	0.89	0.895	0.894	0.897	0.9	0.904	0.905	0.903	0.907	0.908	0.907	0.909	0.909	**0.909**
	5	0.882	0.883	0.888	0.891	0.892	0.899	0.899	0.898	0.902	0.899	0.904	0.903	0.906	0.908	0.905	**0.908**
	7	0.879	0.884	0.889	0.892	0.896	0.897	0.898	0.899	0.899	0.897	0.902	0.904	0.902	0.903	0.907	**0.908**
	9	0.845	0.886	0.891	0.892	0.894	0.897	0.899	0.899	0.901	0.902	0.904	0.904	0.903	0.905	**0.906**	0.905
	11	0.864	0.887	0.886	0.893	0.894	0.895	0.898	0.899	0.9	0.901	0.901	0.904	0.905	0.905	0.904	**0.906**
Dry Bean balanced	3	0.87	0.894	0.898	0.898	0.904	0.902	0.902	0.908	0.904	0.907	0.908	0.906	0.91	0.911	**0.913**	0.911
	5	0.883	0.889	0.888	0.891	0.896	0.9	0.899	0.901	0.905	0.902	0.905	0.906	0.909	0.907	0.908	**0.909**
	7	0.884	0.887	0.886	0.89	0.894	0.898	0.897	0.897	0.901	0.901	0.904	0.901	0.906	0.905	0.904	**0.907**
	9	0.882	0.884	0.889	0.891	0.894	0.896	0.897	0.901	0.899	0.902	0.901	0.902	0.903	0.904	0.906	**0.909**
	11	0.867	0.886	0.885	0.891	0.894	0.893	0.896	0.899	0.898	0.898	0.9	**0.904**	**0.904**	**0.904**	0.903	0.902
Sensorless	3	0.883	0.895	0.899	0.903	0.912	0.909	0.911	0.917	0.913	0.918	0.915	0.914	0.917	0.921	0.918	**0.921**
	5	0.857	0.887	0.898	0.899	0.905	0.903	0.908	0.909	0.913	0.914	0.912	0.916	0.913	0.915	**0.916**	0.913
	7	0.875	0.889	0.895	0.9	0.907	0.908	0.908	0.915	0.913	0.911	0.915	0.915	0.914	0.918	**0.919**	0.915
	9	0.863	0.884	0.893	0.896	0.9	0.905	0.908	0.911	0.91	0.913	0.909	**0.918**	0.917	**0.918**	0.915	0.915
	11	0.833	0.876	0.894	0.893	0.903	0.905	0.909	0.909	0.907	0.909	0.912	0.915	0.917	0.914	**0.918**	0.915
Crowd Sourced imbalanced	3	0.729	0.777	0.766	0.737	0.768	0.8	0.797	0.814	0.813	0.811	0.809	0.8	0.804	**0.827**	0.806	0.809
	5	0.77	0.754	0.759	0.777	0.806	0.822	0.814	0.829	0.831	0.835	0.823	0.821	0.838	**0.84**	0.824	0.838
	7	0.79	0.812	0.808	0.828	0.847	0.83	0.835	0.835	0.848	0.841	0.846	0.844	0.839	0.838	0.838	**0.851**
	9	0.799	0.812	0.833	0.832	0.829	0.842	0.836	0.839	**0.851**	0.842	0.842	0.847	0.824	0.83	0.837	0.841
	11	0.807	0.807	0.827	0.826	**0.846**	0.83	0.833	0.832	0.821	0.829	0.83	0.824	0.838	0.833	0.823	0.838
Crowd Sourced balanced	3	0.658	0.661	0.71	0.723	0.782	0.819	0.786	0.845	0.841	0.842	0.857	0.841	0.871	0.884	0.846	**0.892**
	5	0.677	0.682	0.723	0.78	0.827	0.848	0.849	0.867	0.889	0.888	0.899	0.864	0.89	0.864	**0.914**	0.906
	7	0.703	0.773	0.813	0.838	0.876	0.893	0.886	0.905	0.906	0.919	0.908	0.914	0.916	0.911	0.907	**0.922**
	9	0.778	0.815	0.849	0.869	0.895	0.899	0.905	0.909	0.906	0.903	0.91	0.908	**0.916**	0.913	0.898	0.915
	11	0.805	0.836	0.873	0.883	0.886	0.902	0.895	0.897	0.888	0.891	0.891	0.886	0.892	**0.904**	0.88	0.894

https://doi.org/10.1371/journal.pone.0311041.t002

5 Results and comparisons

The results of the experiments are shown in the tables below. Comparison of experimental results are made in terms of:

- The quality of classification for different numbers of artificial objects created based on one original object in local table.
- The quality of classification for different approaches: average of weights and sum of weights to aggregating local neural networks.

Table 3. Results of classification accuracy *acc* for the proposed approach: One hidden layer, the method to substitute values of missing attributes in local tables—Three artificial objects generated based on one original object, MLP networks aggregation using average of weights and various number of neurons in the hidden layer (3AO-1HL-AVG). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	No. of neurons in hidden layer															
		0.25 × I	0.5 × I	0.75 × I	1 × I	1.5 × I	1.75 × I	2 × I	2.5 × I	2.75 × I	3 × I	3.5 × I	3.75 × I	4 × I	4.5 × I	4.75 × I	5 × I
Vehicle imbalanced	3	0.282	0.592	0.654	0.690	0.672	0.664	0.661	0.685	0.668	0.652	0.675	0.705	0.671	0.689	**0.718**	0.685
	5	0.479	0.588	0.639	0.672	0.664	0.644	0.68	0.69	0.668	0.675	0.664	0.654	0.688	**0.693**	0.664	0.665
	7	0.283	0.538	0.619	0.638	0.676	0.654	0.66	0.672	0.68	**0.686**	0.66	0.685	0.675	0.671	0.675	0.684
	9	0.283	0.661	0.65	0.647	0.681	0.69	0.686	0.682	0.68	0.668	0.65	0.692	0.675	**0.694**	0.684	0.684
	11	0.353	0.615	0.655	0.646	0.672	0.672	0.65	0.642	0.661	0.673	0.675	**0.69**	0.672	0.684	0.66	0.655
Vehicle balanced	3	0.407	0.281	0.681	0.66	0.685	0.686	0.675	0.676	0.722	0.688	0.714	0.701	0.689	0.699	0.718	**0.722**
	5	0.533	0.647	0.652	0.661	0.652	0.657	0.672	0.66	0.693	0.668	0.682	0.698	**0.72**	0.693	0.669	0.659
	7	0.283	0.68	0.668	0.661	0.675	0.707	0.698	0.706	0.709	0.731	0.711	0.714	0.723	**0.732**	0.72	0.728
	9	0.283	0.606	0.539	0.668	0.698	0.68	0.68	0.693	0.69	0.689	0.694	**0.702**	0.692	0.69	0.689	0.697
	11	0.283	0.568	0.648	0.681	0.659	0.673	0.671	0.664	0.677	0.689	0.689	0.685	0.681	**0.692**	0.682	0.686
Dry Bean imbalanced	3	0.87	0.889	0.89	0.898	0.901	0.896	0.901	0.906	0.906	0.907	0.906	0.907	0.909	0.907	0.907	**0.909**
	5	0.88	0.886	0.888	0.893	0.895	0.897	0.895	0.902	0.903	0.903	0.903	0.906	0.908	0.906	**0.91**	0.909
	7	0.879	0.887	0.891	0.89	0.896	0.895	0.897	0.898	0.901	0.901	0.905	0.904	0.906	**0.907**	0.904	0.906
	9	0.832	0.886	0.888	0.892	0.893	0.896	0.897	0.899	0.899	0.899	0.902	0.904	0.904	0.905	**0.906**	0.904
	11	0.839	0.886	0.888	0.889	0.891	0.896	0.897	0.896	0.9	0.899	0.9	0.901	0.902	0.903	0.904	**0.907**
Dry Bean balanced	3	0.887	0.891	0.889	0.897	0.902	0.902	0.904	0.908	0.908	0.907	0.909	0.91	0.909	**0.911**	0.91	**0.911**
	5	0.838	0.888	0.889	0.894	0.897	0.896	0.9	0.902	0.901	0.899	0.904	0.906	0.907	0.906	0.908	**0.91**
	7	0.881	0.882	0.89	0.892	0.895	0.895	0.898	0.897	0.9	0.9	0.902	0.903	0.903	**0.907**	0.905	0.904
	9	0.878	0.884	0.888	0.893	0.891	0.892	0.895	0.899	0.898	0.898	0.904	0.901	0.904	0.904	0.905	**0.906**
	11	0.877	0.886	0.889	0.893	0.895	0.894	0.894	0.896	0.899	0.899	0.9	0.9	0.902	0.902	0.902	**0.905**
Sensorless	3	0.873	0.895	0.903	0.902	0.912	0.909	0.918	0.916	0.908	0.919	0.918	0.92	0.912	0.917	**0.92**	0.918
	5	0.882	0.888	0.902	0.899	0.907	0.906	0.907	0.908	0.911	0.91	0.914	0.909	0.911	0.912	**0.914**	0.912
	7	0.787	0.882	0.89	0.902	0.903	0.906	0.912	0.908	0.911	0.915	0.915	**0.919**	0.916	0.917	0.916	0.916
	9	0.862	0.885	0.888	0.898	0.903	0.907	0.912	0.909	0.911	0.909	0.91	0.909	0.911	**0.916**	0.914	0.914
	11	0.802	0.879	0.89	0.898	0.907	0.907	0.908	0.912	0.91	0.913	0.912	0.915	0.916	**0.922**	0.918	0.917
Crowd Sourced imbalanced	3	0.741	0.74	0.747	0.757	0.754	0.776	0.783	0.8	0.788	0.788	**0.806**	0.791	0.788	0.799	0.796	0.801
	5	0.689	0.752	0.773	0.783	0.811	0.8	0.818	0.818	0.823	0.805	0.81	0.821	0.818	0.808	0.821	**0.827**
	7	0.745	0.818	0.77	0.795	0.813	0.813	0.808	0.833	0.824	0.818	0.831	0.825	0.83	0.823	**0.837**	0.833
	9	0.79	0.779	0.806	0.814	0.834	0.817	0.823	0.829	0.828	0.84	0.839	0.837	0.833	**0.846**	0.843	0.845
	11	0.82	0.825	0.818	0.838	0.842	0.84	0.859	0.855	**0.871**	0.863	0.84	0.857	0.852	0.854	0.857	0.857
Crowd Sourced balanced	3	0.252	0.712	0.755	0.752	0.784	0.75	0.777	0.815	0.776	0.802	0.832	0.819	0.793	0.829	**0.833**	0.821
	5	0.394	0.726	0.685	0.8	0.789	0.82	0.831	0.847	0.843	0.871	0.88	**0.892**	0.879	0.881	0.878	0.884
	7	0.546	0.778	0.799	0.806	0.844	0.879	0.878	0.887	0.883	0.893	0.904	0.911	0.895	0.903	**0.916**	0.916
	9	0.759	0.807	0.788	0.855	0.869	0.878	0.907	0.901	0.9	0.894	**0.921**	0.914	0.917	0.915	0.905	0.895
	11	0.789	0.834	0.83	0.866	0.892	0.891	0.897	0.898	0.904	0.9	0.911	0.912	0.904	0.913	0.907	**0.914**

https://doi.org/10.1371/journal.pone.0311041.t003

- The quality of classification for different number of hidden layers.
- The quality of classification for different number of neurons in the hidden layers.
- The quality of classification of the proposed method versus two other approaches from the literature—homogeneous and heterogeneous ensemble of classifiers.

The average classification accuracy obtained from three runs of the algorithm are presented. Tables 2–5 show the results obtained for one hidden layer, different number of artificial objects

Table 4. Results of classification accuracy *acc* for the proposed approach: One hidden layer, the method to substitute values of missing attributes in local tables—One artificial objects generated based on one original object, MLP networks aggregation using sum of weights and various number of neurons in the hidden layer (1AO-1HL-SUM). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	No. of neurons in hidden layer															
		0.25 × I	0.5 × I	0.75 × I	1 × I	1.5 × I	1.75 × I	2 × I	2.5 × I	2.75 × I	3 × I	3.5 × I	3.75 × I	4 × I	4.5 × I	4.75 × I	5 × I
Vehicle imbalanced	3	0.264	0.571	0.588	0.598	0.625	0.631	0.609	0.609	0.647	**0.675**	0.608	0.644	0.626	0.648	0.631	0.627
	5	0.4	0.626	0.638	0.625	0.602	0.664	0.598	0.66	0.636	0.671	0.646	0.665	0.659	0.661	0.661	**0.673**
	7	0.272	0.643	0.594	0.636	0.671	0.648	0.667	0.626	**0.682**	0.647	0.639	0.673	0.651	0.667	0.647	0.664
	9	0.283	0.496	0.633	0.539	0.646	0.659	0.643	0.655	0.654	0.657	0.669	0.667	**0.697**	0.644	0.685	0.686
	11	0.283	0.65	0.65	0.634	0.633	0.648	0.635	0.631	0.682	0.66	0.684	0.663	0.646	0.671	0.678	**0.694**
Vehicle imbalanced	3	0.461	0.423	0.568	0.64	0.639	0.635	0.661	0.692	0.656	**0.703**	0.66	0.685	0.647	0.686	0.643	0.68
	5	0.283	0.301	0.661	0.627	0.671	0.622	0.654	0.69	0.709	**0.71**	0.678	0.656	0.682	0.69	0.684	0.668
	7	0.283	0.665	0.693	0.655	0.669	0.685	0.707	0.711	0.693	0.682	0.715	0.702	**0.723**	0.701	0.677	0.714
	9	0.286	0.373	0.665	0.642	0.681	0.685	0.643	0.702	0.681	0.668	0.678	0.707	0.702	0.715	0.738	**0.743**
	11	0.283	0.463	0.677	0.615	0.63	0.675	0.675	**0.714**	0.668	0.689	0.706	0.707	0.709	0.709	0.693	0.707
Dry Bean imbalanced	3	0.826	0.891	0.887	0.902	0.901	0.905	0.904	0.909	0.907	0.91	0.91	0.911	0.911	**0.912**	**0.912**	**0.912**
	5	0.878	0.883	0.896	0.904	0.901	0.908	0.908	0.907	0.906	0.909	0.912	0.909	**0.912**	0.911	0.91	0.911
	7	0.858	0.88	0.889	0.9	0.903	0.908	0.907	0.906	0.908	0.908	0.908	0.91	0.911	0.911	0.913	**0.914**
	9	0.898	0.892	0.899	0.902	0.905	0.904	0.907	0.91	0.909	0.909	0.911	0.909	0.913	0.911	**0.914**	0.913
	11	0.877	0.892	0.897	0.901	0.904	0.906	0.907	0.908	0.908	0.91	0.911	0.907	0.91	0.909	**0.913**	0.91
Dry Bean balanced	3	0.809	0.901	0.9	0.905	0.91	0.898	0.907	0.907	0.9	0.909	0.91	0.91	0.909	**0.912**	**0.912**	0.91
	5	0.739	0.89	0.896	0.899	0.907	0.905	0.905	0.906	0.912	0.907	0.908	0.91	0.912	0.91	**0.916**	0.913
	7	0.867	0.892	0.896	0.9	0.904	0.9	0.906	0.907	0.906	0.91	0.911	0.908	0.912	0.911	**0.912**	**0.912**
	9	0.876	0.893	0.894	0.899	0.905	0.901	0.907	0.908	0.91	0.909	0.91	0.911	0.91	**0.913**	0.91	0.911
	11	0.773	0.881	0.895	0.898	0.902	0.904	0.906	0.909	0.904	0.908	0.908	0.911	**0.912**	0.91	0.91	0.911
Sensorless	3	0.864	0.887	0.895	0.901	0.904	0.896	0.899	0.903	0.909	0.906	0.908	0.901	0.916	0.911	0.918	**0.92**
	5	0.869	0.887	0.896	0.899	0.902	0.901	0.904	0.902	0.904	0.904	0.907	0.902	**0.911**	0.908	0.908	0.904
	7	0.869	0.886	0.895	0.895	0.903	0.905	0.902	0.908	0.906	0.91	0.899	0.911	0.909	**0.915**	**0.915**	0.914
	9	0.868	0.895	0.893	0.901	0.902	0.902	0.905	0.906	0.907	0.905	0.909	0.91	0.915	**0.917**	0.914	0.912
	11	0.846	0.893	0.894	0.898	0.903	0.906	0.906	0.908	0.909	0.909	0.912	0.909	0.912	0.911	**0.914**	**0.914**
Crowd Sourced imbalanced	3	0.612	0.756	0.753	0.753	0.757	0.777	0.773	0.782	0.783	0.793	0.792	0.777	0.79	0.793	0.775	**0.796**
	5	0.757	0.705	0.766	0.764	0.783	0.788	0.789	0.795	0.805	0.798	0.8	0.792	0.794	0.807	**0.817**	0.814
	7	0.747	0.733	0.77	0.766	0.763	0.772	0.782	0.798	0.808	0.778	0.814	0.81	0.811	**0.829**	0.803	0.82
	9	0.718	0.758	0.765	0.747	0.774	0.763	0.786	0.768	0.768	0.792	0.807	0.81	0.802	0.804	0.815	**0.832**
	11	0.734	0.758	0.752	0.751	0.764	0.785	0.787	0.806	0.817	0.811	0.815	0.779	0.801	0.803	0.823	**0.827**
Crowd Sourced balanced	3	0.688	0.699	0.695	0.741	0.753	0.794	0.798	0.786	0.817	0.802	0.82	0.814	0.832	0.841	0.849	**0.864**
	5	0.684	0.7	0.649	0.652	0.759	0.75	0.764	0.761	0.767	0.827	0.795	0.823	0.819	0.866	**0.874**	0.851
	7	0.65	0.716	0.72	0.681	0.763	0.769	0.746	0.758	0.776	0.8	0.831	0.799	0.83	**0.864**	0.862	0.839
	9	0.569	0.644	0.697	0.668	0.743	0.714	0.795	0.782	0.789	0.808	0.789	0.815	0.794	**0.86**	0.84	0.812
	11	0.62	0.711	0.663	0.657	0.726	0.77	0.797	0.796	0.812	0.833	0.786	0.789	0.837	0.845	0.844	**0.861**

https://doi.org/10.1371/journal.pone.0311041.t004

(one or three artificial objects generated), different aggregation methods used (average or sum) and different number of neurons in the hidden layer. For simplicity, the designations are adopted:

- 1HL, 2HL—for one or two hidden layers,
- 1AO, 3AO—for one or three generated artificial objects,
- AVG, SUM—for the aggregation method—average and sum.

Table 5. Results of classification accuracy *acc* for the proposed approach: One hidden layer, the method to substitute values of missing attributes in local tables—Three artificial objects generated based on one original object, MLP networks aggregation using sum of weights and various number of neurons in the hidden layer (3AO-1HL-SUM). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	No. of neurons in hidden layer															
		0.25 × I	0.5 × I	0.75 × I	1 × I	1.5 × I	1.75 × I	2 × I	2.5 × I	2.75 × I	3 × I	3.5 × I	3.75 × I	4 × I	4.5 × I	4.75 × I	5 × I
Vehicle imbalanced	3	0.273	0.577	0.598	0.602	0.639	0.644	0.643	0.633	0.651	0.640	0.657	0.636	**0.665**	0.638	0.631	0.643
	5	0.304	0.482	0.591	0.644	0.643	0.639	0.648	0.661	0.663	0.592	0.639	0.634	0.623	0.651	**0.671**	0.635
	7	0.264	0.558	0.585	0.627	0.627	0.626	0.644	0.636	0.654	0.644	0.643	0.685	0.642	**0.696**	0.667	0.657
	9	0.283	0.486	0.636	0.614	0.677	0.663	0.682	0.646	0.701	0.661	**0.717**	0.702	0.701	0.682	0.705	0.675
	11	0.341	0.461	0.647	0.648	0.643	0.655	0.654	0.678	0.681	0.66	0.661	0.669	**0.688**	0.665	0.671	0.657
Vehicle balanced	3	0.386	0.281	0.584	0.543	0.657	0.6	0.627	0.618	0.663	0.648	0.665	0.661	0.623	0.644	**0.677**	0.633
	5	0.353	0.53	0.634	0.629	0.629	0.664	**0.717**	0.643	0.672	0.671	0.64	0.688	0.661	0.667	0.646	0.626
	7	0.264	0.659	0.618	0.609	0.675	0.718	0.699	0.71	0.702	0.709	0.707	**0.735**	0.698	0.701	0.717	0.72
	9	0.283	0.535	0.529	0.693	0.661	0.672	0.681	0.672	**0.711**	0.689	0.702	0.684	0.705	0.698	0.692	0.701
	11	0.283	0.555	0.613	0.638	0.652	0.66	0.68	0.692	0.639	0.693	0.681	0.663	**0.705**	0.701	0.686	0.696
Dry Bean imbalanced	3	0.796	0.895	0.901	0.901	0.904	0.904	0.904	0.909	0.906	0.907	0.909	**0.911**	0.91	0.909	**0.911**	**0.911**
	5	0.859	0.888	0.897	0.9	0.905	0.902	0.905	0.909	0.909	0.908	0.909	0.91	0.91	0.909	**0.913**	**0.913**
	7	0.878	0.889	0.899	0.891	0.906	0.907	0.904	0.91	0.91	0.91	0.909	0.911	0.91	**0.913**	0.911	0.912
	9	0.828	0.885	0.891	0.898	0.901	0.903	0.906	0.906	0.906	0.909	0.91	0.909	0.91	**0.913**	0.912	0.911
	11	0.869	0.887	0.891	0.899	0.902	0.906	0.9	0.907	0.904	0.906	0.907	0.908	**0.911**	0.909	0.909	0.91
Dry Bean balanced	3	0.845	0.884	0.887	0.895	0.905	0.906	0.909	0.91	0.908	0.909	0.907	0.91	**0.911**	**0.911**	0.91	**0.911**
	5	0.815	0.887	0.891	0.903	0.905	0.907	0.906	0.909	0.91	0.909	0.909	0.913	0.911	0.911	0.911	**0.913**
	7	0.864	0.888	0.894	0.901	0.901	0.906	0.908	0.903	0.908	0.906	0.911	0.911	0.909	0.911	**0.913**	0.909
	9	0.863	0.884	0.894	0.896	0.903	0.903	0.907	0.907	0.907	0.908	0.91	0.908	**0.912**	0.91	0.909	**0.912**
	11	0.886	0.885	0.89	0.892	0.9	0.899	0.903	0.905	0.906	0.909	0.907	0.911	0.91	**0.912**	0.911	**0.912**
Sensorless	3	0.795	0.862	0.9	0.887	0.905	0.909	0.911	0.916	**0.918**	0.914	0.915	**0.918**	0.911	0.914	0.917	0.917
	5	0.748	0.885	0.893	0.893	0.899	0.895	0.895	0.903	0.903	0.896	0.903	0.902	0.901	0.9	**0.904**	**0.904**
	7	0.785	0.881	0.868	0.886	0.895	0.896	0.9	0.902	0.898	0.904	0.904	0.905	**0.908**	0.903	0.906	0.902
	9	0.824	0.869	0.882	0.89	0.9	0.898	0.902	0.9	0.902	0.903	0.905	0.906	0.905	**0.908**	0.905	0.907
	11	0.794	0.854	0.858	0.888	0.896	0.892	0.9	0.9	0.903	0.905	0.9	0.907	0.907	0.906	0.908	**0.91**
Crowd Sourced imbalanced	3	0.741	0.74	0.747	0.757	0.754	0.776	0.783	0.8	0.788	0.788	**0.806**	0.791	0.788	0.799	0.796	0.801
	5	0.679	0.737	0.732	0.761	0.776	0.752	0.795	0.783	0.786	0.782	0.789	0.794	0.793	0.785	0.795	**0.802**
	7	0.74	0.753	0.738	0.756	0.748	0.773	0.777	0.795	0.789	0.77	0.794	0.798	0.812	0.814	**0.818**	0.802
	9	0.731	0.725	0.745	0.771	0.736	0.775	0.776	0.779	0.786	0.794	0.797	0.794	**0.808**	0.798	0.798	0.806
	11	0.716	0.77	0.759	0.759	0.781	0.793	0.767	0.799	0.784	0.798	0.804	0.805	0.82	0.826	0.818	**0.842**
Crowd Sourced balanced	3	0.26	0.648	0.714	0.699	0.741	0.722	0.731	0.757	0.742	0.74	**0.782**	0.776	0.756	0.768	0.779	0.776
	5	0.568	0.712	0.724	0.762	0.751	0.793	0.777	0.798	0.779	0.824	0.826	0.832	0.818	0.839	0.822	**0.839**
	7	0.646	0.724	0.741	0.748	0.759	0.793	0.818	0.803	0.81	0.841	0.827	0.843	0.856	0.851	**0.866**	0.857
	9	0.692	0.697	0.728	0.767	0.766	0.775	0.799	0.809	0.78	0.819	0.812	0.84	**0.855**	0.844	0.853	0.853
	11	0.672	0.699	0.722	0.729	0.747	0.767	0.768	0.797	0.817	0.828	0.85	0.834	0.856	0.838	**0.863**	0.821

https://doi.org/10.1371/journal.pone.0311041.t005

Tables 6–21, show the results obtained for two hidden layers and also different number of artificial objects (one or three artificial objects generated), different aggregation methods used (mean and sum) and different number of neurons in the hidden layers. The columns show the number of neurons in the first hidden layer while the rows indicate the different number of neurons in the second hidden layer. The results obtained for different data sets are divided into separate tables due to the size of these tables. In all of these tables, the best result obtained for a given number of hidden layers, a number artificial objects and given aggregation method

Table 6. Results of classification accuracy *acc* for the proposed approach and the Vehicle data sets: Two hidden layers, the method to substitute values of missing attributes in local tables—One artificial objects generated based on one original object, MLP networks aggregation using average of weights and various number of neurons in the hidden layer (1AO-2HL-AVG). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	Second HL	No. of neurons in the first hidden layer															
			0.25 × I	0.5 × I	0.75 × I	1 × I	1.5 × I	1.75 × I	2 × I	2.5 × I	2.75 × I	3 × I	3.5 × I	3.75 × I	4 × I	4.5 × I	4.75 × I	5 × I
Vehicle imbalanced	3	1 × I	0.504	0.635	0.651	0.64	0.668	0.686	0.678	0.673	0.671	0.677	0.675	0.665	0.681	0.693	0.598	0.615
		2 × I	0.612	0.667	0.676	0.676	0.69	0.681	0.706	0.684	0.696	0.698	0.699	0.694	0.727	0.682	0.69	0.682
		3 × I	0.625	0.692	0.686	0.694	0.701	0.692	0.702	0.672	0.71	0.711	0.693	0.696	0.71	0.688	0.718	0.669
		4 × I	0.478	0.702	0.686	0.68	0.703	0.697	0.689	**0.739**	0.692	0.692	0.698	0.703	0.693	0.703	0.689	0.696
		5 × I	0.283	0.68	0.664	0.675	0.675	0.684	0.707	0.714	0.684	0.713	0.726	0.706	0.714	0.667	0.734	0.711
	5	1 × I	0.625	0.636	0.639	0.657	0.621	0.657	0.663	0.677	0.646	0.673	0.671	0.636	0.664	0.657	0.669	0.671
		2 × I	0.409	0.663	0.643	0.65	0.684	0.678	0.692	0.696	0.656	0.68	0.692	0.668	0.672	0.659	0.65	0.664
		3 × I	0.634	0.694	0.648	0.68	0.642	0.681	0.643	0.669	0.673	0.692	0.684	0.669	0.678	0.689	0.664	0.661
		4 × I	0.584	0.664	0.684	0.677	0.671	0.688	**0.709**	0.705	0.665	0.693	0.671	0.671	0.688	0.684	0.677	0.692
		5 × I	0.667	0.665	0.671	0.663	0.69	0.678	0.692	0.654	0.665	0.709	0.665	0.68	0.682	0.69	0.644	0.694
	7	1 × I	0.543	0.648	0.651	0.646	0.639	0.664	0.655	0.657	0.652	0.664	0.634	0.678	0.675	0.655	0.656	0.661
		2 × I	0.588	0.647	0.66	0.639	0.677	0.672	0.673	0.677	0.657	0.669	0.647	0.699	0.664	0.644	0.686	0.655
		3 × I	0.601	0.655	0.664	0.651	0.669	0.669	0.693	0.688	0.675	0.703	0.671	0.68	0.676	0.688	0.685	**0.724**
		4 × I	0.283	0.664	0.685	0.68	0.702	0.693	0.663	0.693	0.705	0.696	0.69	0.711	0.701	0.694	0.685	0.686
		5 × I	0.554	0.665	0.693	0.66	0.671	0.684	0.685	0.682	0.669	0.68	0.705	0.686	0.718	0.69	0.698	0.697
	9	1 × I	0.583	0.633	0.651	0.663	0.68	0.669	0.647	0.623	0.661	0.668	0.671	0.657	0.652	0.681	0.65	0.675
		2 × I	0.587	0.63	0.642	0.663	0.684	0.675	0.668	0.664	0.688	0.659	0.68	0.685	0.675	0.664	0.696	0.668
		3 × I	0.333	0.646	0.668	0.665	0.673	0.688	0.648	0.698	0.682	0.686	0.675	0.684	0.669	**0.713**	0.675	0.688
		4 × I	0.564	0.677	0.66	0.639	0.678	0.686	0.685	0.693	0.681	0.685	0.681	0.701	0.698	0.682	0.685	0.678
		5 × I	0.63	0.652	0.663	0.684	0.693	0.684	0.665	0.676	0.705	0.681	0.689	0.69	0.692	0.696	0.696	0.675
	11	1 × I	0.526	0.622	0.562	0.621	0.627	0.634	0.613	0.633	0.655	0.648	0.647	0.664	0.615	0.629	0.655	0.634
		2 × I	0.283	0.631	0.622	0.635	0.619	0.642	0.642	0.657	0.675	0.682	0.676	0.668	0.659	0.656	0.681	0.65
		3 × I	0.6	0.562	0.583	0.659	0.686	0.678	0.644	0.657	0.648	0.664	0.651	0.696	0.682	0.664	0.685	0.644
		4 × I	0.581	0.657	0.654	0.696	0.678	0.673	0.656	0.664	0.684	0.664	0.685	0.677	0.672	0.692	0.656	0.664
		5 × I	0.584	0.65	0.675	0.605	0.671	0.68	0.664	0.678	0.66	0.678	0.673	0.669	**0.706**	0.697	0.678	0.673
Vehicle balanced	3	1 × I	0.659	0.675	0.678	0.663	0.699	0.642	0.705	0.675	0.684	0.703	0.682	0.651	0.661	0.688	0.686	0.711
		2 × I	0.648	0.635	0.709	0.707	0.682	0.709	0.713	0.698	0.73	0.706	0.718	0.693	0.713	0.707	0.707	0.693
		3 × I	0.63	0.696	0.707	0.718	0.722	0.724	0.731	**0.736**	0.714	0.711	0.707	0.702	0.714	0.717	0.731	0.724
		4 × I	0.609	0.58	0.719	0.732	0.711	0.694	0.717	0.726	0.709	0.711	0.709	0.724	0.699	0.718	0.722	0.703
		5 × I	0.563	0.705	0.697	0.718	0.71	0.719	0.703	0.714	0.706	0.723	0.718	0.723	0.73	0.73	0.711	0.713
	5	1 × I	0.66	0.673	0.664	0.657	0.684	0.697	0.702	0.688	0.678	0.669	0.664	0.706	0.707	0.702	0.715	0.696
		2 × I	0.606	0.681	0.696	0.657	0.709	0.713	0.701	0.693	0.71	0.692	0.705	0.705	0.714	0.677	0.71	0.698
		3 × I	0.577	0.668	0.713	0.696	0.724	0.703	0.693	0.688	0.686	0.685	0.72	0.706	0.707	0.702	0.696	0.699
		4 × I	0.613	0.723	0.696	0.696	0.681	0.702	0.711	**0.728**	0.706	0.697	0.688	0.707	0.717	0.711	0.684	0.72
		5 × I	0.283	0.722	0.706	0.685	0.711	0.714	0.69	0.703	0.719	0.707	0.681	0.723	0.706	0.713	0.694	0.703
	7	1 × I	0.48	0.644	0.705	0.646	0.703	0.676	0.694	0.692	0.714	0.698	0.719	0.699	0.699	0.726	0.694	0.699
		2 × I	0.518	0.718	0.717	0.706	0.697	0.722	0.717	0.684	0.705	0.705	0.717	0.713	0.711	0.682	0.684	0.699
		3 × I	0.374	0.702	0.701	0.715	0.703	0.702	0.709	0.684	0.728	0.723	0.715	0.719	0.714	0.717	0.72	0.713
		4 × I	0.705	0.705	0.71	0.686	0.718	0.741	0.727	0.719	0.714	0.728	0.726	0.743	0.711	0.705	0.73	0.714
		5 × I	0.681	0.706	0.703	0.713	0.715	0.713	0.711	0.72	0.739	0.72	0.73	0.696	0.748	**0.757**	0.715	0.713
	9	1 × I	0.61	0.626	0.669	0.646	0.676	0.659	0.681	0.667	0.643	0.698	0.675	0.682	0.652	0.688	0.677	0.692
		2 × I	0.623	0.625	0.675	0.655	0.678	0.684	0.677	0.673	0.685	0.681	0.709	0.693	0.672	0.681	0.688	0.696
		3 × I	0.677	0.66	0.672	0.698	0.709	0.69	0.667	0.71	0.68	0.707	0.707	0.684	0.678	0.689	0.714	**0.719**
		4 × I	0.589	0.646	0.699	0.684	0.68	0.678	0.664	0.678	0.673	0.689	0.682	0.681	0.705	0.681	0.692	0.682
		5 × I	0.675	0.554	0.705	0.68	0.681	0.703	0.707	0.69	0.678	0.696	0.711	0.697	0.709	0.709	0.697	0.699
	11	1 × I	0.584	0.593	0.579	0.659	0.655	0.587	0.648	0.644	0.685	0.663	0.685	0.646	0.665	0.678	0.688	0.655
		2 × I	0.559	0.617	0.678	0.654	0.678	0.675	0.673	0.694	0.669	0.685	0.69	0.677	0.697	0.685	0.698	0.709
		3 × I	0.526	0.634	0.672	0.669	0.668	0.692	0.698	0.688	0.697	0.686	0.682	0.702	0.702	0.702	0.692	0.685
		4 × I	0.659	0.593	0.66	0.672	0.685	0.681	0.692	0.703	0.714	0.714	0.711	0.709	0.706	0.703	0.719	0.697
		5 × I	0.568	0.581	0.654	0.688	0.696	0.699	0.715	0.71	**0.723**	0.717	0.702	0.682	0.707	0.682	0.718	0.709

https://doi.org/10.1371/journal.pone.0311041.t006

Table 7. Results of classification accuracy *acc* for the proposed approach and the Dry Bean data sets: Two hidden layer, the method to substitute values of missing attributes in local tables—One artificial objects generated based on one original object, MLP networks aggregation using average of weights and various number of neurons in the hidden layer (1AO-2HL-AVG). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	Second HL	No. of neurons in the first hidden layer															
			0.25 × I	0.5 × I	0.75 × I	1 × I	1.5 × I	1.75 × I	2 × I	2.5 × I	2.75 × I	3 × I	3.5 × I	3.75 × I	4 × I	4.5 × I	4.75 × I	5 × I
Dry Bean imbalanced	3	1 × I	0.888	0.891	0.892	0.892	0.893	0.891	0.892	0.894	0.894	0.892	0.896	0.894	0.895	0.895	0.898	0.9
		2 × I	0.891	0.89	0.892	0.893	0.894	0.898	0.894	0.895	0.898	0.896	0.898	0.899	0.9	0.899	0.899	0.896
		3 × I	0.89	0.894	0.893	0.894	0.896	0.896	0.896	0.898	0.899	0.896	0.899	0.902	0.899	0.899	0.901	0.903
		4 × I	0.89	0.894	0.897	0.894	0.897	0.895	0.898	0.896	0.897	0.901	0.903	0.9	0.899	0.899	0.901	**0.906**
		5 × I	0.896	0.894	0.896	0.897	0.896	0.895	0.904	0.901	0.901	0.9	0.905	0.899	0.903	0.902	0.903	0.903
	5	1 × I	0.855	0.888	0.89	0.889	0.89	0.894	0.892	0.896	0.893	0.894	0.894	0.897	0.894	0.894	0.895	0.896
		2 × I	0.82	0.887	0.89	0.892	0.893	0.893	0.893	0.895	0.895	0.896	0.894	0.897	0.896	0.897	0.898	0.894
		3 × I	0.89	0.895	0.895	0.893	0.896	0.895	0.897	0.897	0.9	0.897	0.898	0.897	0.896	0.895	**0.902**	0.9
		4 × I	0.89	0.889	0.897	0.895	0.893	0.898	0.894	0.899	0.896	0.895	0.898	0.897	0.898	0.901	0.898	0.898
		5 × I	0.89	0.891	0.895	0.895	0.896	0.896	0.899	0.897	0.898	0.9	0.899	0.898	0.9	0.901	0.9	**0.902**
	7	1 × I	0.882	0.886	0.887	0.891	0.892	0.891	0.889	0.893	0.892	0.894	0.894	0.893	0.893	0.896	0.893	0.896
		2 × I	0.881	0.889	0.89	0.895	0.896	0.895	0.892	0.894	0.897	0.892	0.896	0.899	0.896	0.897	0.895	0.896
		3 × I	0.859	0.891	0.894	0.891	0.895	0.896	0.897	0.9	0.897	0.896	0.898	0.896	0.898	0.897	0.901	0.895
		4 × I	0.888	0.894	0.893	0.895	0.897	0.893	0.895	0.898	0.899	0.896	0.897	0.897	0.898	0.9	0.899	0.899
		5 × I	0.889	0.893	0.895	0.9	0.896	0.898	0.898	0.9	0.897	0.896	0.898	0.901	0.902	0.899	**0.903**	0.899
	9	1 × I	0.643	0.723	0.8	0.887	0.885	0.887	0.84	0.884	0.889	0.893	0.832	0.839	0.894	0.892	0.891	0.892
		2 × I	0.675	0.885	0.887	0.89	0.834	0.894	0.896	0.896	0.898	0.887	0.886	0.896	0.894	0.9	0.897	0.898
		3 × I	0.763	0.886	0.884	0.862	0.853	0.876	0.888	0.888	0.897	0.893	0.897	0.897	0.898	0.9	0.901	0.897
		4 × I	0.796	0.876	0.874	0.893	0.895	0.892	0.897	0.898	0.899	0.897	0.899	0.9	0.899	0.899	0.902	**0.904**
		5 × I	0.882	0.888	0.882	0.877	0.899	0.895	0.893	0.9	0.9	0.9	0.9	0.902	0.902	0.903	0.9	0.903
	11	1 × I	0.6	0.609	0.815	0.88	0.789	0.889	0.808	0.858	0.893	0.889	0.848	0.888	0.88	0.871	0.891	0.893
		2 × I	0.721	0.819	0.818	0.835	0.863	0.889	0.847	0.853	0.894	0.894	0.898	0.859	0.859	0.897	0.896	0.898
		3 × I	0.79	0.811	0.837	0.884	0.841	0.893	0.853	0.896	0.895	0.881	0.9	0.898	0.898	0.898	0.901	0.899
		4 × I	0.704	0.848	0.843	0.844	0.873	0.887	0.898	0.896	0.895	0.899	0.899	0.886	0.902	**0.903**	0.902	0.902
		5 × I	0.816	0.886	0.857	0.893	0.891	0.897	0.897	0.899	0.899	0.893	0.898	0.9	0.901	0.899	0.9	0.901
Dry Bean balanced	3	1 × I	0.887	0.889	0.892	0.897	0.894	0.893	0.891	0.896	0.895	0.897	0.898	0.896	0.895	0.901	0.898	0.901
		2 × I	0.885	0.891	0.894	0.893	0.896	0.9	0.896	0.899	0.897	0.898	0.899	0.901	0.9	0.899	0.908	0.899
		3 × I	0.886	0.894	0.895	0.892	0.897	0.894	0.896	0.9	0.899	0.901	0.898	0.899	0.906	0.899	0.902	0.9
		4 × I	0.888	0.895	0.893	0.902	0.899	0.908	0.905	0.901	0.897	0.899	0.908	0.899	0.902	0.897	0.905	0.907
		5 × I	0.891	0.894	0.898	0.899	0.897	0.904	0.9	**0.912**	0.898	0.902	0.906	0.904	0.909	0.902	0.905	0.906
	5	1 × I	0.86	0.886	0.891	0.891	0.892	0.89	0.895	0.892	0.893	0.894	0.893	0.893	0.897	0.896	0.893	0.895
		2 × I	0.888	0.891	0.891	0.894	0.895	0.895	0.895	0.896	0.898	0.9	0.896	0.893	0.896	0.895	0.898	0.897
		3 × I	0.891	0.893	0.894	0.896	0.894	0.897	0.897	0.895	0.897	0.897	0.892	0.897	0.897	0.899	0.895	0.895
		4 × I	0.889	0.89	0.892	0.894	0.897	0.898	0.896	0.897	0.898	0.898	0.897	0.899	0.898	0.894	0.9	0.896
		5 × I	0.889	0.892	0.897	0.897	0.898	0.895	0.897	0.897	0.898	**0.901**	0.895	0.899	0.9	0.898	0.898	0.899
	7	1 × I	0.867	0.88	0.888	0.891	0.89	0.89	0.891	0.892	0.894	0.892	0.894	0.893	0.894	0.894	0.894	0.894
		2 × I	0.808	0.886	0.89	0.892	0.898	0.895	0.895	0.897	0.894	0.893	0.895	0.898	0.897	0.897	0.897	0.901
		3 × I	0.881	0.893	0.892	0.891	0.897	0.896	0.898	0.895	0.897	0.898	0.898	0.897	0.899	0.899	0.899	0.899
		4 × I	0.882	0.893	0.893	0.894	0.9	0.897	0.896	0.896	0.897	0.9	0.9	0.898	0.898	**0.903**	0.899	0.897
		5 × I	0.888	0.895	0.891	0.898	0.894	0.9	0.901	0.899	0.899	0.9	0.897	0.901	**0.903**	0.9	0.899	0.899
	9	1 × I	0.753	0.638	0.844	0.784	0.857	0.887	0.849	0.888	0.893	0.89	0.891	0.893	0.894	0.896	0.893	0.897
		2 × I	0.867	0.851	0.891	0.883	0.881	0.896	0.897	0.893	0.867	0.897	0.898	0.898	0.895	0.898	0.896	0.9
		3 × I	0.797	0.887	0.891	0.889	0.893	0.878	0.896	0.895	0.896	0.899	0.9	0.897	0.895	0.901	0.899	0.896
		4 × I	0.808	0.888	0.875	0.894	0.899	0.894	0.902	0.897	0.899	0.901	0.898	0.899	0.9	**0.903**	0.9	0.902
		5 × I	0.847	0.886	0.886	0.896	0.896	0.896	0.9	0.897	0.902	0.897	0.899	0.901	0.901	0.901	0.901	0.9
	11	1 × I	0.733	0.633	0.822	0.883	0.885	0.877	0.824	0.893	0.882	0.893	0.871	0.877	0.891	0.894	0.896	0.895
		2 × I	0.588	0.749	0.887	0.86	0.889	0.892	0.891	0.893	0.894	0.893	0.895	0.895	0.896	0.895	0.896	0.899
		3 × I	0.839	0.882	0.85	0.865	0.887	0.885	0.891	0.877	0.897	0.896	0.899	0.896	0.899	0.899	0.897	0.899
		4 × I	0.759	0.836	0.847	0.896	0.894	0.892	0.895	0.895	0.896	0.899	0.897	0.897	0.897	0.898	**0.901**	0.899
		5 × I	0.825	0.889	0.888	0.889	0.894	0.873	0.898	0.897	0.9	0.9	0.898	0.899	0.899	0.897	0.9	0.899

https://doi.org/10.1371/journal.pone.0311041.t007

Table 8. Results of classification accuracy *acc* for the proposed approach and the Sensorless data sets: Two hidden layer, the method to substitute values of missing attributes in local tables—One artificial objects generated based on one original object, MLP networks aggregation using average of weights and various number of neurons in the hidden layer (1AO-2HL-AVG). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	Second HL	No. of neurons in the first hidden layer															
			0.25 × I	**0.5 × I**	**0.75 × I**	**1 × I**	**1.5 × I**	**1.75 × I**	**2 × I**	**2.5 × I**	**2.75 × I**	**3 × I**	**3.5 × I**	**3.75 × I**	**4 × I**	**4.5 × I**	**4.75 × I**	**5 × I**
Sensorless imbalanced	3	1 × I	0.895	0.92	0.92	0.933	0.923	0.93	0.925	0.937	0.93	0.935	0.938	0.933	0.936	0.936	0.938	0.933
		2 × I	0.915	0.932	0.93	0.936	0.94	0.937	0.938	0.939	0.942	0.938	0.942	0.942	0.944	0.935	0.94	0.939
		3 × I	0.928	0.933	0.937	0.936	0.943	0.94	0.942	0.944	0.943	0.944	0.942	0.939	0.943	0.943	0.945	0.942
		4 × I	0.926	0.93	0.933	0.943	0.94	0.942	0.944	0.947	0.949	0.945	0.944	0.94	0.943	0.942	0.942	0.945
		5 × I	0.927	0.94	0.935	0.944	0.946	0.944	0.945	**0.95**	0.944	0.945	0.948	0.945	0.942	0.943	0.943	0.944
	5	1 × I	0.867	0.903	0.902	0.916	0.919	0.92	0.925	0.923	0.93	0.925	0.921	0.93	0.931	0.926	0.927	0.928
		2 × I	0.909	0.916	0.921	0.924	0.927	0.927	0.93	0.934	0.933	0.935	0.936	0.931	0.935	0.938	0.936	0.934
		3 × I	0.909	0.919	0.92	0.934	0.932	0.929	0.938	0.937	0.938	0.936	0.94	0.941	0.939	0.94	0.939	0.939
		4 × I	0.917	0.92	0.935	0.932	0.936	0.941	0.936	0.937	0.945	0.945	0.938	0.941	0.94	0.94	0.942	0.94
		5 × I	0.912	0.923	0.928	0.939	0.935	0.941	0.94	0.937	0.938	0.942	0.944	0.941	**0.945**	0.94	0.937	0.937
	7	1 × I	0.829	0.892	0.905	0.911	0.914	0.92	0.919	0.925	0.922	0.925	0.925	0.923	0.925	0.93	0.927	0.929
		2 × I	0.9	0.916	0.918	0.92	0.925	0.928	0.928	0.929	0.932	0.933	0.932	0.934	0.937	0.939	0.938	0.935
		3 × I	0.911	0.919	0.921	0.926	0.938	0.929	0.934	0.934	0.938	0.939	0.943	0.94	0.938	0.939	0.943	0.941
		4 × I	0.917	0.922	0.927	0.929	0.941	0.935	0.937	0.935	0.942	0.938	0.933	0.946	0.935	0.939	0.941	**0.947**
		5 × I	0.913	0.927	0.927	0.941	0.928	0.934	0.939	0.942	0.945	0.941	0.941	0.944	0.943	0.938	0.937	0.94
	9	1 × I	0.875	0.901	0.91	0.908	0.923	0.915	0.919	0.92	0.915	0.923	0.917	0.925	0.928	0.92	0.925	0.918
		2 × I	0.907	0.912	0.921	0.921	0.921	0.924	0.933	0.924	0.93	0.93	0.928	0.929	0.931	0.934	0.932	0.932
		3 × I	0.913	0.916	0.928	0.925	0.927	0.924	0.941	0.932	0.94	0.932	0.936	0.936	0.942	0.939	0.935	0.934
		4 × I	0.916	0.922	0.931	0.923	0.932	0.939	0.94	0.937	0.935	0.935	0.939	0.933	0.936	0.939	0.942	0.934
		5 × I	0.923	0.921	0.927	0.934	0.931	0.934	0.934	0.939	0.941	0.932	0.936	0.937	0.942	0.938	**0.943**	0.935
	11	1 × I	0.844	0.903	0.916	0.912	0.908	0.914	0.923	0.915	0.927	0.925	0.917	0.927	0.924	0.914	0.929	0.928
		2 × I	0.901	0.911	0.912	0.914	0.927	0.924	0.922	0.924	0.927	0.93	0.932	0.925	0.926	0.933	0.925	0.925
		3 × I	0.905	0.92	0.922	0.925	0.922	0.929	0.937	0.934	0.93	0.931	0.936	0.932	0.933	0.934	0.926	0.936
		4 × I	0.909	0.919	0.926	0.932	0.93	0.934	0.937	0.935	0.939	0.938	0.931	0.934	0.935	0.938	**0.942**	0.935
		5 × I	0.917	0.926	0.929	0.927	0.936	0.933	0.936	0.937	0.929	0.938	0.932	0.931	0.933	0.941	0.939	0.935

https://doi.org/10.1371/journal.pone.0311041.t008

Table 9. Results of classification accuracy *acc* for the proposed approach and the Crowd Sourced data sets: Two hidden layer, the method to substitute values of missing attributes in local tables—One artificial objects generated based on one original object, MLP networks aggregation using average of weights and various number of neurons in the hidden layer (1AO-2HL-AVG). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	Second HL	No. of neurons in hidden layer															
			0.25 × I	0.5 × I	0.75 × I	1 × I	1.5 × I	1.75 × I	2 × I	2.5 × I	2.75 × I	3 × I	3.5 × I	3.75 × I	4 × I	4.5 × I	4.75 × I	5 × I
Crowd Sourced imbalanced	3	1 × I	0.706	0.765	0.782	0.79	0.782	0.794	0.797	0.809	0.827	0.827	0.821	0.82	0.822	0.812	0.821	0.824
		2 × I	0.754	0.777	0.777	0.791	0.806	0.819	0.806	0.814	0.827	0.828	0.817	0.812	0.812	0.827	0.813	0.83
		3 × I	0.759	0.777	0.815	0.779	0.813	0.805	0.792	0.818	0.823	0.827	0.82	0.818	0.814	0.818	0.822	0.824
		4 × I	0.734	0.81	0.807	0.791	0.804	0.802	0.806	0.822	0.816	0.831	0.822	0.813	0.809	0.836	**0.838**	0.815
		5 × I	0.759	0.807	0.811	0.798	0.799	0.801	0.822	0.823	0.807	0.795	0.827	0.828	0.814	0.817	0.823	0.832
	5	1 × I	0.832	0.84	0.838	0.838	0.85	0.851	0.859	0.856	0.857	**0.875**	0.86	0.863	0.872	0.864	0.852	0.851
		2 × I	0.823	0.842	0.83	0.85	0.843	0.853	0.847	0.86	0.843	0.859	0.863	0.851	0.846	0.847	0.861	0.855
		3 × I	0.823	0.822	0.849	0.841	0.853	0.84	0.847	0.839	0.844	0.832	0.85	0.847	0.847	0.854	0.864	0.846
		4 × I	0.801	0.825	0.835	0.841	0.842	0.85	0.851	0.842	0.854	0.833	0.866	0.853	0.838	0.847	0.839	0.838
		5 × I	0.808	0.83	0.842	0.839	0.833	0.846	0.846	0.818	0.853	0.859	0.834	0.847	0.834	0.855	0.833	0.87
	7	1 × I	0.817	0.823	0.818	0.845	0.846	0.835	**0.855**	0.847	0.835	0.839	0.854	0.84	0.853	0.837	0.846	0.85
		2 × I	0.816	0.834	0.835	0.851	0.824	0.833	0.841	0.849	0.849	0.846	0.843	0.847	0.847	0.834	0.841	0.843
		3 × I	0.817	0.825	0.837	0.824	0.832	0.846	0.842	0.841	0.832	0.832	0.838	0.833	0.85	0.835	0.844	0.831
		4 × I	0.798	0.8	0.811	0.826	0.837	0.843	0.806	0.843	0.842	0.834	0.848	0.845	0.825	0.839	0.839	0.826
		5 × I	0.8	0.813	0.818	0.809	0.839	0.824	0.814	0.821	0.83	0.826	0.846	0.836	0.836	0.836	0.843	0.829
	9	1 × I	0.808	0.829	0.835	0.835	0.852	0.847	0.84	0.837	0.833	0.835	0.842	0.841	0.847	0.843	0.852	0.845
		2 × I	0.81	0.827	0.841	0.845	0.835	0.858	0.835	0.84	0.84	0.844	0.846	0.846	0.843	0.842	**0.859**	0.848
		3 × I	0.815	0.824	0.83	0.838	0.835	0.855	0.853	0.839	0.848	0.841	0.844	0.848	0.853	0.842	0.846	0.855
		4 × I	0.813	0.826	0.826	0.833	0.839	0.848	0.845	0.847	0.853	0.847	0.849	0.858	0.84	0.842	0.851	0.847
		5 × I	0.797	0.803	0.811	0.823	0.843	0.835	0.837	0.834	0.849	0.832	0.833	0.845	0.83	0.822	0.838	0.845
	11	1 × I	0.829	0.829	0.845	0.851	0.854	0.854	0.847	0.858	0.863	0.864	0.856	0.859	0.856	0.862	0.856	0.862
		2 × I	0.817	0.842	0.833	0.859	0.862	0.846	0.849	0.855	0.862	0.843	0.851	0.851	0.86	0.857	0.863	0.855
		3 × I	0.802	0.835	0.829	0.837	0.847	0.851	0.862	0.865	0.861	0.865	0.859	0.85	0.854	0.86	0.857	0.853
		4 × I	0.826	0.822	0.842	0.855	0.842	0.857	0.847	0.848	0.858	0.853	**0.876**	0.862	0.857	0.861	0.86	0.865
		5 × I	0.806	0.833	0.849	0.852	0.847	0.85	0.854	0.855	0.859	0.849	0.864	0.862	0.865	0.841	0.864	0.85
Crowd Sourced balanced	3	1 × I	0.675	0.749	0.793	0.821	0.776	0.831	0.86	0.85	0.858	0.875	0.875	0.882	0.874	0.893	0.869	0.874
		2 × I	0.66	0.774	0.784	0.844	0.845	0.851	0.819	0.868	0.893	0.877	0.888	0.87	0.873	0.894	0.902	0.903
		3 × I	0.634	0.753	0.795	0.827	0.837	0.861	0.874	0.854	0.861	0.889	0.901	0.898	0.86	0.907	0.9	0.911
		4 × I	0.523	0.763	0.83	0.805	0.872	0.864	0.874	0.876	0.869	0.888	0.891	0.905	0.898	0.896	0.897	0.911
		5 × I	0.702	0.786	0.787	0.818	0.871	0.872	0.874	0.887	0.892	0.893	0.898	0.892	0.898	0.905	0.89	**0.915**
	5	1 × I	0.821	0.849	0.867	0.882	0.892	0.901	0.909	0.908	0.91	0.915	0.91	0.905	0.913	0.918	0.9	0.907
		2 × I	0.812	0.863	0.887	0.894	0.902	0.908	0.898	0.912	0.912	0.915	0.919	0.92	0.917	0.917	**0.927**	0.913
		3 × I	0.822	0.88	0.896	0.895	0.913	0.904	0.908	0.914	0.912	0.914	0.906	0.916	0.92	0.917	0.917	0.916
		4 × I	0.845	0.878	0.898	0.889	0.9	0.906	0.909	0.916	0.907	0.917	0.911	0.917	0.915	0.913	0.918	0.915
		5 × I	0.838	0.869	0.895	0.899	0.899	0.911	0.908	0.909	0.909	0.91	0.921	0.922	0.916	0.914	0.917	0.909
	7	1 × I	0.829	0.867	0.885	0.9	0.898	0.905	0.91	0.912	**0.918**	0.908	0.908	0.909	0.905	0.91	0.913	**0.918**
		2 × I	0.836	0.878	0.896	0.894	0.895	0.906	0.899	0.896	0.913	0.902	0.902	0.903	0.909	0.896	0.898	0.91
		3 × I	0.842	0.879	0.892	0.885	0.89	0.895	0.885	0.898	0.909	0.875	0.888	0.903	0.895	0.885	0.901	0.89
		4 × I	0.837	0.891	0.889	0.885	0.892	0.888	0.891	0.895	0.874	0.89	0.884	0.894	0.894	0.89	0.86	0.892
		5 × I	0.85	0.866	0.89	0.882	0.875	0.884	0.898	0.896	0.898	0.891	0.89	0.881	0.882	0.883	0.874	0.898
	9	1 × I	0.838	0.878	0.885	0.89	0.903	0.904	0.891	0.911	0.907	0.895	0.896	0.896	0.9	0.91	0.867	0.905
		2 × I	0.836	0.87	0.893	0.888	0.89	0.906	0.898	0.902	0.886	0.888	0.913	0.87	0.894	**0.915**	0.912	0.912
		3 × I	0.825	0.835	0.869	0.869	0.896	0.898	0.869	0.874	0.9	0.907	0.89	0.893	0.897	0.886	0.879	0.862
		4 × I	0.831	0.852	0.882	0.881	0.865	0.888	0.88	0.883	0.896	0.886	0.892	0.878	0.892	0.879	0.899	0.898
		5 × I	0.854	0.856	0.847	0.869	0.883	0.817	0.85	0.886	0.883	0.828	0.872	0.895	0.839	0.904	0.877	0.895
	11	1 × I	0.826	0.879	0.888	0.894	0.901	0.9	0.894	0.902	0.896	0.891	**0.921**	0.913	0.918	0.908	0.917	0.92
		2 × I	0.844	0.875	0.901	0.887	0.897	0.896	0.9	0.878	0.89	0.883	0.904	0.897	0.906	0.904	0.903	0.913
		3 × I	0.822	0.864	0.875	0.874	0.887	0.9	0.877	0.875	0.905	0.864	0.91	0.875	0.884	0.898	0.868	0.901
		4 × I	0.846	0.851	0.885	0.863	0.853	0.899	0.887	0.89	0.898	0.861	0.904	0.905	0.893	0.887	0.915	0.837
		5 × I	0.835	0.863	0.84	0.869	0.831	0.87	0.853	0.898	0.883	0.854	0.874	0.878	0.848	0.909	0.903	0.917

https://doi.org/10.1371/journal.pone.0311041.t009

Table 10. Results of classification accuracy *acc* for the proposed approach and the Vehicle data sets: Two hidden layers, the method to substitute values of missing attributes in local tables—Three artificial objects generated based on one original object, MLP networks aggregation using average of weights and various number of neurons in the hidden layer (3AO-2HL-AVG). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	Second HL	No. of neurons in hidden layer															
			0.25 × I	0.5 × I	0.75 × I	1 × I	1.5 × I	1.75 × I	2 × I	2.5 × I	2.75 × I	3 × I	3.5 × I	3.75 × I	4 × I	4.5 × I	4.75 × I	5 × I
Vehicle imbalanced	3	1 × I	0.281	0.638	0.669	0.642	0.668	0.689	0.702	0.677	0.69	0.68	0.675	0.675	0.64	0.654	0.677	0.675
		2 × I	0.545	0.659	0.677	0.686	0.69	0.709	0.657	0.685	0.714	0.697	0.705	0.684	0.668	0.696	0.678	0.698
		3 × I	0.563	0.714	0.701	0.714	0.701	0.688	0.689	0.715	0.703	0.705	0.688	0.697	0.717	0.703	0.698	0.692
		4 × I	0.535	0.694	0.722	0.689	**0.732**	0.692	0.715	0.719	0.711	0.696	0.688	0.713	0.717	0.705	0.731	0.702
		5 × I	0.588	0.655	0.702	0.707	0.702	0.678	0.722	0.697	0.707	0.709	0.706	0.693	0.711	0.69	0.697	0.709
	5	1 × I	0.472	0.594	0.66	0.681	0.682	0.66	0.66	0.677	0.659	0.677	0.68	0.673	0.671	0.669	0.671	0.689
		2 × I	0.28	0.661	0.676	0.665	0.667	0.676	0.669	0.686	0.648	0.689	0.673	0.69	0.677	0.685	0.686	0.678
		3 × I	0.283	0.629	0.694	0.696	0.705	0.696	0.692	0.686	0.66	0.699	0.694	0.681	0.678	0.672	0.699	**0.711**
		4 × I	0.505	0.655	0.701	0.636	0.686	0.703	0.69	0.668	0.672	0.673	0.665	0.693	0.689	0.682	0.696	0.685
		5 × I	0.564	0.68	0.673	0.685	0.698	0.685	0.684	0.671	0.701	0.667	0.706	0.692	0.688	0.694	0.669	0.688
	7	1 × I	0.58	0.639	0.638	0.652	0.631	0.65	0.657	0.675	0.647	0.654	0.665	0.629	0.654	0.65	0.643	0.638
		2 × I	0.579	0.626	0.651	0.659	0.65	0.667	0.696	0.663	0.688	0.664	0.689	0.661	0.65	0.657	0.642	0.676
		3 × I	0.617	0.681	0.63	0.685	0.699	0.671	0.693	0.671	0.663	0.699	**0.702**	0.681	0.685	0.677	0.692	0.688
		4 × I	0.593	0.66	0.676	0.69	0.684	0.682	0.69	0.677	0.678	0.697	0.692	0.684	0.682	0.654	0.659	0.694
		5 × I	0.543	0.667	0.686	0.612	0.681	0.68	0.693	0.654	0.692	0.686	0.693	0.693	0.689	0.701	0.699	0.692
	9	1 × I	0.537	0.496	0.613	0.633	0.612	0.559	0.626	0.605	0.617	0.629	0.587	0.646	0.634	0.65	0.638	0.614
		2 × I	0.587	0.598	0.64	0.656	0.639	0.598	0.661	0.671	0.625	0.638	0.612	0.642	0.656	0.652	0.643	0.627
		3 × I	0.517	0.572	0.651	0.635	0.668	0.64	0.639	0.65	0.669	0.668	0.643	0.652	0.65	0.669	0.63	0.643
		4 × I	0.322	0.587	0.623	0.659	0.634	0.686	0.629	0.682	0.655	0.64	0.638	0.659	0.663	0.631	0.659	0.68
		5 × I	0.52	0.642	0.621	0.655	0.647	0.639	0.654	0.652	0.657	0.63	0.661	0.667	0.676	0.639	0.646	**0.686**
	11	1 × I	0.385	0.597	0.6	0.652	0.623	0.646	0.608	0.631	0.629	0.667	0.65	0.646	0.661	0.619	0.659	0.64
		2 × I	0.585	0.638	0.646	0.642	0.635	0.657	0.659	0.668	0.66	0.667	0.657	0.667	0.625	0.663	0.667	0.669
		3 × I	0.581	0.589	0.663	0.646	0.64	0.614	0.65	0.621	0.656	0.685	0.692	0.676	0.66	0.678	0.646	0.678
		4 × I	0.606	0.66	0.614	0.669	0.659	0.671	0.681	0.677	0.669	**0.698**	0.657	0.676	0.68	0.675	0.66	0.686
		5 × I	0.529	0.65	0.678	0.657	0.677	0.676	0.685	0.68	0.669	0.689	0.671	0.684	0.68	0.696	0.667	0.684
Vehicle blanced	3	1 × I	0.571	0.684	0.69	0.724	0.72	0.713	0.73	0.709	0.693	0.703	0.715	0.699	0.707	0.759	0.726	0.283
		2 × I	0.703	0.702	0.726	0.739	0.724	0.718	0.739	0.743	0.752	0.731	0.736	0.731	0.732	0.73	0.734	0.718
		3 × I	0.698	0.715	0.702	0.745	0.734	0.728	0.722	0.738	0.719	0.736	0.73	0.732	0.727	0.711	0.747	0.745
		4 × I	0.49	0.752	0.757	0.727	0.731	0.723	0.726	0.719	0.752	0.726	**0.76**	0.745	0.743	0.751	0.74	0.743
		5 × I	0.669	0.681	0.73	0.722	0.724	0.739	0.702	0.732	0.744	0.722	0.74	0.736	0.72	0.724	0.749	0.744
	5	1 × I	0.381	0.694	0.626	0.698	0.686	0.688	0.707	0.663	0.682	0.696	0.71	0.714	0.686	0.693	0.697	0.689
		2 × I	0.542	0.669	0.722	0.685	0.697	0.707	0.719	0.715	0.706	0.703	0.728	0.709	0.717	0.707	0.703	0.69
		3 × I	0.594	0.698	0.68	0.698	0.697	0.717	0.706	0.713	**0.732**	0.711	0.72	0.701	0.72	0.705	0.717	0.711
		4 × I	0.568	0.705	0.72	0.694	0.718	0.703	0.726	0.705	0.703	0.718	0.715	0.73	0.707	0.707	0.715	0.701
		5 × I	0.686	0.711	0.714	0.714	0.709	0.718	0.718	0.698	0.72	0.71	0.703	0.709	0.705	0.706	0.688	0.697
	7	1 × I	0.448	0.587	0.673	0.682	0.682	0.678	0.692	0.692	0.71	0.702	0.709	0.682	0.718	0.688	0.718	0.646
		2 × I	0.605	0.64	0.684	0.672	0.705	0.706	0.705	0.71	0.711	0.689	0.71	0.699	0.719	0.706	0.718	0.735
		3 × I	0.283	0.702	0.684	0.703	0.718	0.707	0.734	0.702	0.709	0.703	0.71	0.677	0.723	0.735	0.718	0.718
		4 × I	0.513	0.672	0.673	0.73	0.731	0.724	0.713	0.728	0.726	0.728	0.699	**0.74**	0.701	0.69	0.726	0.711
		5 × I	0.283	0.735	0.723	0.711	0.724	0.71	0.705	0.711	0.719	0.718	0.722	0.735	0.71	0.693	0.707	0.738
	9	1 × I	0.382	0.6	0.671	0.63	0.47	0.664	0.688	0.668	0.65	0.668	0.697	0.663	0.684	0.681	0.682	0.665
		2 × I	0.563	0.622	0.659	0.692	0.699	0.703	0.685	0.69	0.684	0.707	0.685	0.714	0.703	0.71	0.709	0.705
		3 × I	0.283	0.65	0.659	0.694	0.685	0.699	0.718	0.702	0.684	0.685	0.717	0.726	0.717	0.703	0.669	0.714
		4 × I	0.618	0.629	0.71	0.719	0.698	0.709	0.705	0.672	0.664	0.709	0.705	0.705	0.701	0.676	0.676	0.689
		5 × I	0.64	0.654	0.703	0.685	0.68	0.684	0.673	0.715	0.723	0.705	0.707	0.685	**0.728**	0.696	0.678	0.698
	11	1 × I	0.588	0.566	0.575	0.588	0.64	0.673	0.647	0.66	0.626	0.661	0.656	0.61	0.671	0.655	0.664	0.676
		2 × I	0.577	0.6	0.631	0.661	0.652	0.657	0.655	0.657	0.69	0.686	0.685	0.686	0.698	0.673	0.68	0.682
		3 × I	0.6	0.581	0.657	0.664	0.681	0.675	0.703	0.694	0.699	0.694	0.685	0.685	0.696	0.669	0.709	0.677
		4 × I	0.556	0.642	0.701	0.669	0.694	0.68	0.705	0.675	0.703	0.699	0.684	0.707	0.702	0.696	0.689	0.701
		5 × I	0.629	0.602	0.678	0.696	0.682	0.693	0.684	**0.71**	0.672	0.661	0.69	0.689	0.69	0.707	0.69	0.686

https://doi.org/10.1371/journal.pone.0311041.t010

Table 11. Results of classification accuracy *acc* for the proposed approach and the Dry Bean data sets: Two hidden layers, the method to substitute values of missing attributes in local tables—Three artificial objects generated based on one original object, MLP networks aggregation using average of weights and various number of neurons in the hidden layer (3AO-2HL-AVG). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	Second HL	No. of neurons in hidden layer															
			0.25 × I	0.5 × I	0.75 × I	1 × I	1.5 × I	1.75 × I	2 × I	2.5 × I	2.75 × I	3 × I	3.5 × I	3.75 × I	4 × I	4.5 × I	4.75 × I	5 × I
Dry Bean imbalanced	3	1 × I	0.888	0.892	0.893	0.894	0.895	0.893	0.895	0.894	0.894	0.897	0.895	0.899	0.898	0.898	0.899	0.895
		2 × I	0.89	0.893	0.894	0.892	0.895	0.896	0.895	0.893	0.897	0.896	0.895	0.897	0.897	0.897	0.899	0.902
		3 × I	0.891	0.892	0.894	0.895	0.896	0.897	0.899	0.896	0.901	0.897	0.899	0.899	0.902	0.899	0.904	0.906
		4 × I	0.891	0.894	0.897	0.897	0.897	0.898	0.899	0.895	0.899	0.9	0.899	0.9	0.902	0.898	0.904	0.903
		5 × I	0.893	0.892	0.892	0.898	0.9	0.899	0.899	0.9	0.901	0.904	0.897	0.903	0.904	**0.91**	0.902	0.902
	5	1 × I	0.88	0.882	0.888	0.889	0.893	0.895	0.891	0.892	0.893	0.896	0.896	0.896	0.897	0.892	0.895	0.894
		2 × I	0.891	0.888	0.892	0.893	0.896	0.893	0.895	0.895	0.897	0.896	0.897	0.898	0.896	0.895	0.896	0.899
		3 × I	0.89	0.891	0.896	0.896	0.895	0.896	0.896	0.897	0.897	0.896	0.896	0.895	0.898	0.896	0.898	0.899
		4 × I	0.894	0.894	0.895	0.898	0.896	0.9	0.896	0.898	0.899	0.894	0.897	0.898	0.893	0.897	0.899	0.899
		5 × I	0.892	0.895	0.894	0.898	0.896	0.897	0.898	0.897	0.897	0.899	0.896	0.899	0.897	**0.901**	0.899	0.898
	7	1 × I	0.651	0.888	0.866	0.885	0.891	0.894	0.89	0.894	0.894	0.892	0.894	0.893	0.893	0.895	0.892	0.895
		2 × I	0.887	0.889	0.89	0.888	0.893	0.894	0.893	0.892	0.897	0.896	0.896	0.895	0.899	0.896	0.899	0.896
		3 × I	0.89	0.886	0.893	0.893	0.896	0.896	0.895	0.897	0.895	0.896	0.897	0.895	0.899	0.899	0.894	0.895
		4 × I	0.88	0.895	0.894	0.893	0.896	0.897	0.897	0.894	0.898	0.897	0.897	0.9	0.897	0.897	0.899	0.9
		5 × I	0.889	0.889	0.893	0.897	0.892	0.895	0.895	0.896	0.902	0.896	0.897	0.899	0.896	0.898	**0.902**	0.899
	9	1 × I	0.631	0.885	0.879	0.886	0.82	0.888	0.889	0.894	0.888	0.888	0.886	0.889	0.894	0.892	0.893	0.893
		2 × I	0.702	0.873	0.866	0.889	0.886	0.889	0.893	0.892	0.896	0.893	0.892	0.899	0.895	0.901	0.896	0.897
		3 × I	0.836	0.836	0.889	0.887	0.893	0.896	0.898	0.897	0.897	0.897	0.898	0.898	0.898	0.899	0.899	0.898
		4 × I	0.886	0.86	0.873	0.892	0.896	0.899	0.899	0.899	0.899	0.899	0.901	0.899	0.897	0.9	0.901	0.901
		5 × I	0.866	0.892	0.889	0.894	0.898	0.897	0.902	0.897	0.9	0.902	0.901	0.898	0.901	0.897	0.9	**0.903**
	11	1 × I	0.642	0.672	0.738	0.823	0.865	0.8	0.885	0.891	0.889	0.843	0.863	0.893	0.851	0.857	0.892	0.893
		2 × I	0.705	0.742	0.831	0.826	0.889	0.847	0.854	0.897	0.892	0.894	0.889	0.889	0.88	0.881	0.894	0.897
		3 × I	0.8	0.835	0.827	0.837	0.855	0.893	0.868	0.894	0.897	0.896	0.898	0.895	0.888	0.901	0.895	0.9
		4 × I	0.757	0.833	0.855	0.875	0.895	0.867	0.895	0.899	0.897	0.897	0.899	0.876	0.902	0.902	0.9	0.9
		5 × I	0.831	0.84	0.888	0.87	0.854	0.896	0.871	0.896	0.899	0.896	0.873	0.899	0.902	0.9	**0.903**	**0.903**
Dry Bean blanced	3	1 × I	0.889	0.889	0.892	0.895	0.896	0.894	0.894	0.897	0.896	0.897	0.898	0.902	0.902	0.9	0.899	0.903
		2 × I	0.894	0.892	0.895	0.896	0.899	0.899	0.903	0.9	0.904	0.9	0.905	0.898	0.903	0.899	0.9	0.901
		3 × I	0.892	0.894	0.894	0.898	0.897	0.899	0.903	0.899	0.901	0.905	0.903	0.903	0.904	0.907	0.907	0.906
		4 × I	0.897	0.895	0.896	0.903	0.896	0.899	0.901	0.901	0.903	0.904	0.898	0.901	0.907	0.91	0.906	0.903
		5 × I	0.89	0.895	0.896	0.896	0.899	0.9	0.901	0.909	0.907	0.904	0.904	0.899	0.91	0.905	**0.918**	0.907
	5	1 × I	0.888	0.89	0.889	0.89	0.894	0.893	0.893	0.894	0.895	0.895	0.896	0.896	0.895	0.895	0.895	0.899
		2 × I	0.885	0.894	0.891	0.893	0.897	0.893	0.897	0.895	0.894	0.898	0.897	0.898	0.897	0.897	0.893	0.9
		3 × I	0.886	0.889	0.895	0.897	0.894	0.896	0.894	0.897	0.895	0.899	0.896	0.898	0.896	0.899	0.899	0.9
		4 × I	0.893	0.893	0.897	0.894	0.895	0.893	0.897	0.898	0.898	0.898	0.897	0.898	**0.901**	0.896	**0.901**	**0.901**
		5 × I	0.889	0.893	0.894	0.895	0.895	0.9	0.9	0.9	0.9	0.899	0.9	0.899	0.897	0.899	0.899	0.9
	7	1 × I	0.593	0.882	0.889	0.886	0.891	0.895	0.892	0.895	0.894	0.892	0.894	0.893	0.891	0.895	0.895	0.894
		2 × I	0.885	0.85	0.892	0.893	0.893	0.897	0.892	0.895	0.896	0.896	0.896	0.895	0.898	0.896	0.898	0.897
		3 × I	0.878	0.888	0.891	0.89	0.895	0.892	0.894	0.896	0.896	0.898	0.897	0.897	0.896	0.898	0.901	0.898
		4 × I	0.882	0.892	0.895	0.893	0.896	0.897	0.898	0.898	0.9	0.894	0.895	0.898	0.895	0.896	0.898	0.897
		5 × I	0.891	0.895	0.897	0.898	0.897	0.897	0.899	0.897	0.899	0.899	0.899	0.898	0.897	0.9	0.901	**0.902**
	9	1 × I	0.688	0.842	0.733	0.829	0.871	0.89	0.885	0.89	0.893	0.888	0.891	0.889	0.897	0.894	0.891	0.895
		2 × I	0.681	0.818	0.886	0.893	0.891	0.881	0.893	0.895	0.893	0.897	0.896	0.896	0.898	0.898	0.9	0.896
		3 × I	0.88	0.836	0.888	0.842	0.895	0.899	0.895	0.897	0.896	0.9	0.899	0.897	0.899	0.898	0.897	0.9
		4 × I	0.766	0.891	0.865	0.893	0.897	0.871	0.897	0.9	0.895	0.899	0.899	0.902	0.903	**0.904**	0.902	0.901
		5 × I	0.855	0.892	0.886	0.897	0.9	0.897	0.897	0.9	0.9	0.901	0.903	0.901	0.9	0.9	0.903	0.901
	11	1 × I	0.671	0.869	0.88	0.795	0.892	0.888	0.892	0.853	0.889	0.886	0.845	0.893	0.897	0.895	0.896	0.892
		2 × I	0.686	0.882	0.886	0.839	0.892	0.871	0.893	0.89	0.894	0.895	0.895	0.896	0.861	0.895	0.9	0.895
		3 × I	0.813	0.837	0.85	0.89	0.892	0.897	0.894	0.882	0.899	0.9	0.899	0.895	0.898	0.897	0.898	0.899
		4 × I	0.829	0.888	0.889	0.891	0.895	0.897	0.898	0.899	0.9	0.887	0.901	0.902	0.903	0.899	0.899	0.901
		5 × I	0.808	0.836	0.879	0.882	0.894	0.899	0.896	0.898	**0.904**	0.9	0.9	0.898	0.901	0.902	0.9	0.902

https://doi.org/10.1371/journal.pone.0311041.t011

Table 12. Results of classification accuracy *acc* for the proposed approach and the Sensorless data sets: Two hidden layer, the method to substitute values of missing attributes in local tables—Three artificial objects generated based on one original object, MLP networks aggregation using average of weights and various number of neurons in the hidden layer (3AO-2HL-AVG). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	Second HL	No. of neurons in hidden layer															
			0.25 × I	0.5 × I	0.75 × I	1 × I	1.5 × I	1.75 × I	2 × I	2.5 × I	2.75 × I	3 × I	3.5 × I	3.75 × I	4 × I	4.5 × I	4.75 × I	5 × I
Sensorless imbalanced	3	1 × I	0.892	0.921	0.926	0.926	0.92	0.929	0.938	0.933	0.934	0.935	0.93	0.936	0.937	0.942	0.936	0.938
		2 × I	0.921	0.928	0.935	0.94	0.939	0.942	0.938	0.938	0.941	0.934	0.937	0.94	0.948	0.945	0.944	0.94
		3 × I	0.932	0.936	0.94	0.944	0.941	0.948	0.944	0.941	0.948	0.946	0.943	0.943	0.945	0.942	0.945	0.943
		4 × I	0.935	0.951	0.942	0.945	0.945	0.949	0.948	0.944	0.949	0.946	0.945	0.947	0.948	0.941	0.934	0.94
		5 × I	0.941	0.94	0.949	0.946	0.944	0.949	**0.951**	0.948	0.947	0.948	0.941	0.945	0.944	0.945	0.944	0.945
	5	1 × I	0.886	0.902	0.912	0.928	0.922	0.923	0.92	0.925	0.934	0.927	0.926	0.935	0.932	0.929	0.934	0.928
		2 × I	0.907	0.916	0.923	0.923	0.924	0.929	0.927	0.934	0.933	0.933	0.936	0.933	0.94	0.934	0.935	0.934
		3 × I	0.923	0.922	0.928	0.931	0.937	0.937	0.938	0.937	0.939	0.935	0.935	0.94	0.94	0.942	0.942	0.938
		4 × I	0.918	0.927	0.936	0.935	0.939	0.938	0.937	0.939	0.938	0.941	0.935	0.94	0.941	0.94	0.937	0.939
		5 × I	0.924	0.931	0.931	0.932	0.938	0.943	0.94	0.936	0.937	**0.944**	0.942	0.94	0.942	0.942	0.943	0.941
	7	1 × I	0.894	0.911	0.913	0.912	0.914	0.919	0.917	0.92	0.917	0.927	0.926	0.92	0.924	0.926	0.929	0.929
		2 × I	0.909	0.913	0.924	0.923	0.925	0.929	0.928	0.935	0.931	0.924	0.936	0.927	0.934	0.934	0.933	0.932
		3 × I	0.916	0.918	0.927	0.925	0.936	0.929	0.934	0.93	0.936	0.933	0.937	0.935	0.934	0.941	0.935	0.932
		4 × I	0.917	0.927	0.929	0.927	0.934	0.935	0.934	0.932	0.93	0.935	0.937	0.938	0.937	0.942	0.942	0.932
		5 × I	0.914	0.926	0.931	0.933	0.939	0.937	0.937	0.938	0.94	0.942	0.939	0.938	0.939	**0.943**	0.941	0.941
	9	1 × I	0.874	0.902	0.908	0.916	0.916	0.909	0.923	0.929	0.918	0.918	0.921	0.917	0.926	0.933	0.928	0.926
		2 × I	0.909	0.919	0.922	0.921	0.922	0.922	0.928	0.926	0.93	0.923	0.927	0.932	0.929	0.92	0.932	0.931
		3 × I	0.912	0.916	0.925	0.926	0.931	0.937	0.927	0.929	0.93	0.929	0.935	0.932	0.935	0.932	0.938	0.934
		4 × I	0.912	0.93	0.924	0.933	0.934	0.932	0.936	0.935	0.931	0.933	0.936	0.937	0.932	0.931	0.929	0.933
		5 × I	0.918	0.934	0.933	0.932	0.933	0.927	0.94	0.934	0.934	**0.94**	0.936	0.939	0.932	0.936	0.931	0.938
	11	1 × I	0.884	0.903	0.901	0.916	0.92	0.926	0.921	0.923	0.927	0.922	0.918	0.924	0.922	0.929	0.921	0.928
		2 × I	0.9	0.909	0.927	0.924	0.926	0.918	0.923	0.922	0.926	0.923	0.921	0.926	0.923	0.925	0.93	0.93
		3 × I	0.913	0.916	0.922	0.922	0.924	0.927	0.929	0.924	0.931	0.927	0.932	0.93	0.929	0.937	0.928	0.931
		4 × I	0.912	0.915	0.931	0.922	0.933	0.939	0.935	0.931	0.932	0.925	0.926	0.933	0.934	0.927	0.936	0.926
		5 × I	0.917	0.927	0.933	0.93	0.937	0.931	0.935	0.938	**0.939**	0.927	0.93	0.932	0.93	0.925	0.934	0.932

https://doi.org/10.1371/journal.pone.0311041.t012

Table 13. Results of classification accuracy *acc* for the proposed approach and the Crowd Sourced data sets: Two hidden layers, the method to substitute values of missing attributes in local tables—Three artificial objects generated based on one original object, MLP networks aggregation using average of weights and various number of neurons in the hidden layer (3AO-2HL-AVG). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	Second HL	No. of neurons in hidden layer															
			0.25 × I	0.5 × I	0.75 × I	1 × I	1.5 × I	1.75 × I	2 × I	2.5 × I	2.75 × I	3 × I	3.5 × I	3.75 × I	4 × I	4.5 × I	4.75 × I	5 × I
Crowd Sourced imbalanced	3	1 × I	0.717	0.762	0.755	0.748	0.773	0.796	0.756	0.788	0.772	0.806	0.788	0.808	0.801	0.825	0.8	0.805
		2 × I	0.651	0.759	0.763	0.77	0.799	0.8	0.764	0.782	0.788	0.796	0.819	0.816	0.817	0.819	0.813	0.791
		3 × I	0.7	0.769	0.8	0.749	0.787	0.794	0.798	0.808	0.793	0.803	0.82	0.794	0.796	0.808	0.822	**0.833**
		4 × I	0.688	0.733	0.765	0.794	0.782	0.8	0.823	0.807	0.818	0.798	0.821	0.823	0.806	0.802	0.809	0.79
		5 × I	0.7	0.758	0.764	0.794	0.794	0.796	0.784	0.781	0.792	0.814	0.79	0.803	0.813	0.802	0.825	0.83
	5	1 × I	0.8	0.823	0.812	0.841	0.847	0.853	0.839	0.853	0.855	0.852	0.862	0.863	0.857	**0.869**	0.846	0.86
		2 × I	0.805	0.823	0.814	0.848	0.845	0.84	0.863	0.848	0.854	0.855	0.862	0.851	0.852	0.858	0.853	0.848
		3 × I	0.819	0.847	0.794	0.845	0.844	0.851	0.861	0.847	0.86	0.858	0.861	0.86	0.857	0.863	0.851	0.862
		4 × I	0.811	0.821	0.827	0.845	0.844	0.853	0.847	0.861	0.855	0.859	0.86	0.863	0.856	0.859	0.859	0.86
		5 × I	0.8	0.834	0.823	0.845	0.832	0.851	0.848	0.857	0.847	0.851	0.853	0.852	0.853	0.858	0.865	0.862
	7	1 × I	0.794	0.841	0.833	0.849	0.859	0.852	0.855	0.859	0.85	0.848	0.851	0.86	0.846	0.861	0.854	0.847
		2 × I	0.822	0.83	0.841	0.842	0.852	0.852	0.858	0.852	0.858	0.846	0.866	0.854	0.855	0.848	0.864	0.856
		3 × I	0.822	0.819	0.854	0.847	0.858	0.86	0.863	0.858	0.849	0.847	0.847	0.847	0.84	0.852	**0.867**	0.853
		4 × I	0.799	0.825	0.836	0.841	0.834	0.853	0.86	0.854	0.849	0.859	0.853	0.854	0.852	0.865	0.853	0.853
		5 × I	0.821	0.847	0.851	0.843	0.856	0.853	0.841	0.846	0.849	0.857	0.847	0.846	0.849	0.846	0.847	0.851
	9	1 × I	0.823	0.825	0.832	0.845	0.85	0.841	0.844	0.854	0.847	0.846	0.841	0.851	0.839	0.849	0.84	0.837
		2 × I	0.817	0.824	0.839	0.829	0.846	0.837	0.841	0.845	0.847	0.858	0.844	0.853	0.846	0.843	0.845	0.852
		3 × I	0.803	0.829	0.836	0.848	0.835	0.84	0.83	0.835	0.855	0.841	0.839	0.842	0.85	0.839	0.837	0.849
		4 × I	0.782	0.818	0.834	0.842	0.847	0.825	0.841	0.831	0.839	0.845	0.845	0.836	0.843	0.84	0.85	0.854
		5 × I	0.798	0.814	0.833	0.835	0.835	**0.859**	0.843	0.845	0.853	0.839	0.835	0.834	0.837	0.847	0.844	0.843
	11	1 × I	0.824	0.828	0.835	0.835	0.837	0.833	0.842	0.841	0.834	0.854	0.85	0.844	0.847	**0.861**	0.841	0.852
		2 × I	0.814	0.833	0.847	0.843	0.84	0.84	0.84	0.842	0.852	0.85	0.853	0.841	0.847	0.848	0.847	0.846
		3 × I	0.819	0.827	0.84	0.836	0.846	0.831	0.841	0.841	0.852	0.834	0.851	0.841	0.844	0.847	0.85	0.85
		4 × I	0.816	0.814	0.83	0.824	0.842	0.842	0.847	0.848	0.84	0.836	0.843	0.849	0.85	0.846	0.85	0.844
		5 × I	0.804	0.822	0.814	0.83	0.829	0.843	0.851	0.843	0.848	0.841	0.847	0.838	0.858	0.848	0.847	0.85
Crowd Sourced balanced	3	1 × I	0.609	0.689	0.671	0.732	0.76	0.762	0.768	0.779	0.772	0.837	0.799	0.755	0.806	0.821	0.733	0.803
		2 × I	0.683	0.649	0.734	0.753	0.737	0.746	0.787	0.759	0.78	0.793	0.807	0.81	**0.858**	0.786	0.819	0.803
		3 × I	0.639	0.68	0.667	0.696	0.713	0.804	0.796	0.806	0.781	0.807	0.788	0.805	0.828	0.763	0.831	0.854
		4 × I	0.573	0.602	0.732	0.701	0.739	0.687	0.767	0.596	0.816	0.788	0.765	0.829	0.824	0.82	0.791	0.806
		5 × I	0.662	0.727	0.653	0.74	0.766	0.733	0.761	0.76	0.804	0.812	0.727	0.757	0.833	0.778	0.807	0.851
	5	1 × I	0.711	0.824	0.839	0.854	0.851	0.891	0.87	0.899	0.895	0.875	0.885	0.911	0.901	0.908	0.905	**0.919**
		2 × I	0.611	0.832	0.83	0.853	0.875	0.892	0.893	0.888	0.905	0.905	0.867	0.906	0.905	0.899	0.91	0.909
		3 × I	0.721	0.822	0.817	0.874	0.866	0.899	0.898	0.898	0.89	0.907	0.893	0.895	0.911	0.909	0.907	0.907
		4 × I	0.736	0.858	0.868	0.854	0.89	0.903	0.883	0.904	0.892	0.866	0.91	0.911	0.896	0.883	0.915	0.915
		5 × I	0.763	0.856	0.875	0.881	0.896	0.886	0.897	0.889	0.907	0.891	0.898	0.898	0.91	0.915	0.899	0.91
	7	1 × I	0.809	0.864	0.879	0.888	0.899	0.902	0.887	0.907	0.916	0.911	0.917	0.906	0.917	0.913	0.916	0.91
		2 × I	0.825	0.882	0.876	0.891	0.894	0.892	0.896	0.912	0.91	0.912	0.913	0.91	0.902	0.917	0.917	0.915
		3 × I	0.823	0.872	0.868	0.889	0.911	0.909	0.901	0.911	0.911	0.913	0.908	0.913	0.904	0.908	0.917	0.914
		4 × I	0.832	0.853	0.888	0.895	0.909	0.907	0.908	0.916	0.913	0.912	0.902	0.913	0.91	0.916	0.916	0.914
		5 × I	0.843	0.877	0.894	0.903	0.906	0.901	0.903	0.906	0.9	0.907	0.891	0.911	0.915	**0.918**	0.914	0.912
	9	1 × I	0.785	0.848	0.874	0.897	0.899	0.905	0.906	0.904	0.907	0.909	0.902	0.912	0.909	0.912	0.913	0.914
		2 × I	0.837	0.863	0.893	0.898	0.903	0.901	0.91	0.913	0.906	0.906	0.897	0.912	0.91	0.9	0.902	**0.916**
		3 × I	0.826	0.854	0.892	0.883	0.902	0.891	0.89	0.897	0.887	0.9	0.891	0.902	0.898	0.901	0.889	0.914
		4 × I	0.809	0.848	0.887	0.894	0.903	0.892	0.909	0.889	0.897	0.898	0.885	0.899	0.898	0.907	0.904	0.901
		5 × I	0.839	0.852	0.876	0.902	0.884	0.903	0.893	0.891	0.901	0.899	0.91	0.905	0.909	0.886	0.903	0.895
	11	1 × I	0.819	0.845	0.885	0.893	0.875	0.91	0.903	0.908	0.898	0.909	0.905	0.909	0.908	0.913	0.909	0.915
		2 × I	0.826	0.86	0.853	0.899	0.889	0.901	0.89	0.919	0.915	0.908	0.913	**0.923**	0.915	0.897	0.91	0.906
		3 × I	0.834	0.85	0.887	0.887	0.891	0.877	0.907	0.901	0.897	0.908	0.891	0.906	0.905	0.918	0.919	0.921
		4 × I	0.837	0.84	0.873	0.866	0.895	0.875	0.907	0.892	0.893	0.903	0.911	0.911	0.89	0.917	0.916	0.907
		5 × I	0.822	0.85	0.84	0.871	0.885	0.879	0.891	0.889	0.903	0.913	0.911	0.909	0.9	0.909	0.908	0.912

https://doi.org/10.1371/journal.pone.0311041.t013

Table 14. Results of classification accuracy *acc* for the proposed approach and the Vehicle data sets: Two hidden layers, the method to substitute values of missing attributes in local tables—One artificial objects generated based on one original object, MLP networks aggregation using sum of weights and various number of neurons in the hidden layer (1AO-2HL-SUM). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	Second HL	No. of neurons in hidden layer															
			0.25 × I	0.5 × I	0.75 × I	1 × I	1.5 × I	1.75 × I	2 × I	2.5 × I	2.75 × I	3 × I	3.5 × I	3.75 × I	4 × I	4.5 × I	4.75 × I	5 × I
Vehicle imbalanced	3	1 × I	0.283	0.42	0.521	0.556	0.629	0.651	0.673	0.654	0.68	0.661	0.636	0.664	0.652	0.668	0.575	0.285
		2 × I	0.53	0.604	0.663	0.657	0.643	0.659	0.638	0.663	0.647	0.684	0.601	0.69	0.699	0.602	0.635	0.657
		3 × I	0.475	0.684	0.49	0.663	0.689	0.655	0.671	0.671	0.676	0.688	0.652	0.673	0.675	0.664	0.638	0.656
		4 × I	0.411	0.664	0.63	0.639	0.685	0.698	0.654	0.676	0.694	0.647	0.678	0.699	0.681	0.698	0.676	0.686
		5 × I	0.283	0.64	0.6	0.702	0.643	0.657	0.707	0.656	0.66	0.702	0.651	**0.713**	**0.713**	0.664	0.696	0.661
	5	1 × I	0.558	0.598	0.636	0.631	0.593	0.673	0.647	0.65	0.635	0.626	0.385	0.648	0.631	0.663	0.318	0.642
		2 × I	0.28	0.621	0.622	0.642	0.681	0.654	0.652	0.675	0.664	0.677	0.673	0.684	0.68	0.676	0.661	0.663
		3 × I	0.597	0.609	0.66	0.672	0.682	0.673	0.648	0.688	0.673	0.652	0.678	0.665	0.664	0.686	0.684	0.672
		4 × I	0.467	0.614	0.647	0.643	0.709	0.669	0.705	0.684	0.677	0.668	0.672	0.706	0.686	0.671	0.702	0.681
		5 × I	0.642	0.631	0.676	0.654	0.709	0.682	0.639	0.657	0.698	**0.715**	0.698	0.677	0.699	0.656	0.682	0.669
	7	1 × I	0.647	0.618	0.644	0.648	0.692	0.681	0.681	0.706	0.608	0.63	0.36	0.68	0.688	0.693	0.69	0.688
		2 × I	0.613	0.69	0.634	0.696	0.693	0.705	0.693	0.714	0.715	0.713	0.668	0.719	0.705	0.667	**0.735**	0.706
		3 × I	0.516	0.714	0.668	0.698	0.702	0.69	0.702	0.706	0.696	0.705	0.71	0.711	0.715	0.694	0.703	0.698
		4 × I	0.283	0.657	0.693	0.723	0.699	0.706	0.715	0.71	0.693	0.714	0.711	0.693	0.714	0.717	0.722	0.724
		5 × I	0.516	0.684	0.709	0.686	0.703	0.677	0.701	0.714	0.702	0.734	0.731	0.723	0.724	0.728	0.698	0.703
	9	1 × I	0.591	0.676	0.612	0.675	0.731	0.713	0.68	0.692	0.667	0.709	0.623	0.684	0.719	0.402	0.698	0.709
		2 × I	0.623	0.69	0.675	0.718	0.715	0.696	0.694	0.752	0.694	0.705	0.727	0.706	0.711	0.732	0.727	0.718
		3 × I	0.446	0.618	0.69	0.718	0.71	0.728	0.72	0.71	0.74	0.689	0.739	0.709	0.735	0.73	0.715	0.719
		4 × I	0.56	0.688	0.72	0.732	0.693	0.697	0.728	0.747	0.709	0.738	0.718	0.732	0.734	0.724	0.728	0.735
		5 × I	0.634	0.663	0.702	0.711	0.743	0.734	0.738	0.734	0.734	0.723	0.735	0.735	0.697	0.748	**0.752**	0.722
	11	1 × I	0.596	0.652	0.636	0.69	0.69	0.699	0.69	0.698	0.651	0.648	0.697	0.714	0.709	0.656	0.723	0.681
		2 × I	0.283	0.634	0.685	0.698	0.699	0.703	0.709	0.696	0.717	0.727	0.705	0.717	0.684	0.701	0.728	0.693
		3 × I	0.564	0.594	0.604	0.71	0.686	0.677	0.698	0.739	0.689	0.706	0.734	0.72	0.717	0.717	0.713	0.697
		4 × I	0.605	0.693	0.678	0.701	0.705	0.71	0.717	0.723	0.732	0.726	0.689	0.724	0.715	0.701	**0.743**	0.715
		5 × I	0.576	0.623	0.714	0.715	0.715	0.723	0.738	0.73	0.727	0.714	0.732	0.706	0.719	0.69	0.715	0.732
Vehicle blanced	3	1 × I	0.533	0.562	0.587	0.643	0.631	0.623	0.657	0.64	0.722	0.699	0.638	0.692	0.592	0.678	0.692	0.69
		2 × I	0.614	0.555	0.655	0.672	0.677	0.661	0.669	0.681	0.705	0.638	0.663	0.689	0.659	0.701	0.688	0.68
		3 × I	0.518	0.664	0.688	0.643	0.676	0.677	0.682	0.705	0.719	0.702	0.693	0.665	0.681	0.699	0.693	0.646
		4 × I	0.57	0.479	0.676	0.706	0.686	0.652	0.665	0.688	0.693	0.707	0.685	0.672	0.719	0.697	0.696	0.693
		5 × I	0.547	0.671	0.642	0.699	0.659	0.696	0.714	0.677	0.696	**0.726**	0.678	0.667	0.688	0.715	0.699	0.693
	5	1 × I	0.636	0.664	0.651	0.661	0.688	0.654	0.69	0.673	0.698	0.677	0.671	0.707	0.694	0.677	0.694	0.699
		2 × I	0.589	0.672	0.709	0.659	0.697	0.713	0.726	0.707	0.731	0.705	0.715	0.718	0.697	0.713	0.719	0.682
		3 × I	0.52	0.692	0.675	0.69	0.706	0.724	0.703	0.717	0.692	0.738	0.711	0.732	0.735	0.724	0.706	0.719
		4 × I	0.591	0.692	0.702	0.701	0.699	0.688	0.706	0.702	0.686	0.743	0.722	0.719	0.743	0.726	0.702	0.71
		5 × I	0.283	0.672	0.696	0.68	**0.748**	0.726	0.717	0.696	0.724	0.727	0.722	0.696	0.706	0.723	0.719	0.723
	7	1 × I	0.549	0.547	0.709	0.707	0.686	0.702	0.731	0.714	0.726	0.73	0.722	0.713	0.743	0.741	0.739	0.718
		2 × I	0.5	0.707	0.71	0.727	0.69	0.724	0.735	0.735	0.723	0.717	0.736	0.727	0.745	0.73	0.726	0.739
		3 × I	0.583	0.702	0.709	0.723	0.723	0.732	0.741	0.71	0.749	0.745	0.738	0.734	0.738	0.749	0.748	0.756
		4 × I	0.694	0.707	0.715	0.707	0.724	0.741	0.743	0.734	0.751	0.72	0.734	0.74	0.734	0.749	0.745	0.751
		5 × I	0.643	0.685	0.719	0.74	0.706	0.738	0.724	0.732	0.741	0.728	0.739	0.743	**0.759**	0.747	0.738	0.743
	9	1 × I	0.618	0.579	0.69	0.656	0.701	0.682	0.694	0.705	0.713	0.723	0.736	0.676	0.714	0.731	0.686	0.731
		2 × I	0.619	0.612	0.684	0.684	0.697	0.698	0.713	0.719	0.722	0.71	0.734	0.724	0.713	0.719	0.73	0.719
		3 × I	0.609	0.689	0.675	0.689	0.735	0.72	0.732	0.717	0.711	0.724	0.707	0.731	0.724	0.719	0.739	0.731
		4 × I	0.577	0.644	0.692	0.694	0.732	0.709	0.743	0.724	0.738	0.711	**0.745**	0.73	0.726	0.68	0.724	0.734
		5 × I	0.589	0.56	0.715	0.713	0.717	0.731	0.74	0.718	0.723	0.73	0.713	0.728	0.736	0.707	0.707	0.705
	11	1 × I	0.581	0.657	0.692	0.711	0.724	0.689	0.723	0.699	0.736	0.71	0.719	0.697	0.692	0.709	0.726	0.654
		2 × I	0.626	0.65	0.71	0.692	0.717	0.741	0.72	0.709	0.718	0.732	0.73	0.73	0.738	0.715	0.751	0.719
		3 × I	0.556	0.686	0.726	0.73	0.723	0.705	0.743	0.735	0.734	0.759	0.761	0.726	0.731	0.724	0.701	0.759
		4 × I	0.618	0.703	0.702	0.751	0.709	0.724	0.752	0.745	0.757	0.734	0.757	0.755	**0.768**	0.723	0.738	0.761
		5 × I	0.528	0.672	0.71	0.734	0.751	0.747	0.73	0.72	0.756	0.751	0.741	0.751	0.738	**0.768**	0.749	0.739

https://doi.org/10.1371/journal.pone.0311041.t014

Table 15. Results of classification accuracy *acc* for the proposed approach and the Dry Bean data sets: Two hidden layers, the method to substitute values of missing attributes in local tables—One artificial objects generated based on one original object, MLP networks aggregation using sum of weights and various number of neurons in the hidden layer (1AO-2HL-SUM). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	Second HL	No. of neurons in the first hidden layer															
			0.25 × I	0.5 × I	0.75 × I	1 × I	1.5 × I	1.75 × I	2 × I	2.5 × I	2.75 × I	3 × I	3.5 × I	3.75 × I	4 × I	4.5 × I	4.75 × I	5 × I
Dry Bean imbalanced	3	1 × I	0.883	0.895	0.901	0.895	0.898	0.906	0.901	0.909	0.905	0.901	0.904	0.913	0.907	0.912	0.907	0.917
		2 × I	0.905	0.89	0.909	0.902	0.909	0.916	0.913	0.915	0.91	0.913	0.911	0.913	0.918	0.915	0.915	0.912
		3 × I	0.901	0.899	0.908	0.907	0.906	0.908	0.909	0.913	0.917	0.913	0.913	0.918	0.914	0.915	0.914	0.916
		4 × I	0.891	0.903	0.907	0.914	0.903	0.91	0.91	0.915	0.914	0.918	**0.92**	0.917	0.915	0.918	0.918	0.92
		5 × I	0.905	0.912	0.908	0.899	0.912	0.915	0.919	0.916	0.915	0.915	**0.92**	0.918	0.919	0.917	0.918	0.916
	5	1 × I	0.887	0.887	0.888	0.895	0.898	0.897	0.902	0.907	0.901	0.896	0.908	0.906	0.906	0.91	0.909	0.909
		2 × I	0.887	0.893	0.903	0.904	0.902	0.898	0.905	0.903	0.906	0.908	0.913	0.912	0.914	0.914	0.91	0.913
		3 × I	0.892	0.889	0.898	0.903	0.91	0.912	0.91	0.913	0.909	0.908	0.913	0.911	0.915	0.912	0.913	0.916
		4 × I	0.881	0.898	0.898	0.905	0.908	0.909	0.91	0.911	0.912	0.913	0.912	0.915	0.913	0.914	0.914	**0.918**
		5 × I	0.899	0.899	0.901	0.901	0.91	0.914	0.907	0.913	0.915	0.913	0.911	0.916	0.913	0.914	0.917	0.916
	7	1 × I	0.895	0.894	0.893	0.896	0.901	0.897	0.895	0.899	0.91	0.903	0.907	0.904	0.899	0.902	0.902	0.914
		2 × I	0.889	0.899	0.896	0.903	0.902	0.902	0.905	0.904	0.899	0.907	0.904	0.911	0.905	0.908	0.904	0.91
		3 × I	0.894	0.895	0.895	0.897	0.905	0.91	0.911	0.913	0.909	0.911	0.91	0.913	0.911	0.912	0.914	0.915
		4 × I	0.894	0.893	0.9	0.903	0.902	0.907	0.905	0.912	0.913	0.911	0.913	0.914	0.912	0.914	0.913	0.915
		5 × I	0.887	0.898	0.897	0.909	0.914	0.909	0.905	0.91	0.911	0.909	0.911	0.916	0.916	0.916	0.914	**0.917**
	9	1 × I	0.885	0.892	0.889	0.889	0.894	0.894	0.898	0.898	0.892	0.898	0.902	0.895	0.899	0.896	0.9	0.901
		2 × I	0.888	0.893	0.895	0.899	0.898	0.895	0.9	0.898	0.901	0.901	0.903	0.899	0.903	0.901	0.908	0.909
		3 × I	0.892	0.897	0.897	0.899	0.897	0.908	0.901	0.904	0.903	0.908	0.905	0.908	0.905	0.909	0.906	0.911
		4 × I	0.89	0.896	0.895	0.9	0.899	0.901	0.902	0.902	0.905	0.905	0.907	0.906	0.912	0.906	0.912	0.907
		5 × I	0.893	0.896	0.899	0.895	0.903	0.898	0.905	0.904	0.909	0.911	0.909	**0.915**	0.912	0.911	0.914	0.913
	11	1 × I	0.897	0.888	0.892	0.891	0.894	0.893	0.894	0.895	0.898	0.895	0.897	0.9	0.897	0.898	0.898	0.904
		2 × I	0.888	0.897	0.896	0.891	0.896	0.899	0.898	0.896	0.902	0.901	0.896	0.899	0.9	0.906	0.898	0.904
		3 × I	0.885	0.894	0.892	0.9	0.897	0.899	0.901	0.899	0.899	0.907	0.905	0.903	0.899	0.899	0.906	0.903
		4 × I	0.889	0.891	0.896	0.899	0.898	0.9	0.901	0.905	0.902	0.903	0.907	0.902	0.905	0.898	0.907	0.906
		5 × I	0.89	0.895	0.895	0.893	0.897	0.903	0.901	0.902	0.906	0.902	0.906	0.902	0.909	**0.914**	0.908	0.907
Dry Bean blanced	3	1 × I	0.889	0.888	0.899	0.908	0.908	0.905	0.901	0.912	0.913	0.907	0.912	0.91	0.911	0.912	0.914	0.912
		2 × I	0.884	0.899	0.902	0.907	0.911	0.912	0.911	0.907	0.911	0.915	0.906	0.917	0.916	0.916	0.916	0.918
		3 × I	0.871	0.893	0.909	0.901	0.91	0.909	0.908	0.914	0.917	0.914	0.91	0.913	0.914	0.916	0.916	0.916
		4 × I	0.902	0.909	0.913	0.913	0.915	0.915	0.912	0.917	0.915	0.915	0.914	0.916	0.917	0.917	0.917	0.918
		5 × I	0.891	0.91	0.915	0.907	0.916	0.916	0.913	0.919	0.917	0.916	0.916	0.915	0.917	0.917	**0.919**	0.916
	5	1 × I	0.889	0.888	0.897	0.905	0.903	0.909	0.903	0.912	0.902	0.908	0.91	0.906	0.91	0.907	0.913	0.911
		2 × I	0.893	0.896	0.898	0.905	0.905	0.91	0.911	0.91	0.913	0.908	0.915	0.914	0.909	0.915	0.913	0.916
		3 × I	0.909	0.903	0.902	0.902	0.91	0.909	0.912	0.914	0.914	**0.917**	0.911	0.913	0.915	**0.917**	0.912	0.916
		4 × I	0.887	0.893	0.902	0.904	0.908	0.912	0.911	0.914	0.914	0.913	0.916	0.914	0.915	0.914	0.916	0.914
		5 × I	0.892	0.893	0.909	0.906	0.915	0.912	0.915	0.914	**0.917**	0.915	0.915	**0.917**	**0.917**	0.915	**0.917**	0.916
	7	1 × I	0.887	0.895	0.895	0.889	0.899	0.895	0.897	0.898	0.904	0.901	0.903	0.904	0.904	0.911	0.904	0.905
		2 × I	0.893	0.892	0.893	0.899	0.908	0.899	0.905	0.91	0.901	0.911	0.906	0.911	0.91	0.911	0.909	0.907
		3 × I	0.894	0.892	0.893	0.904	0.902	0.906	0.909	0.905	0.908	0.913	0.911	0.908	0.914	0.914	0.912	0.912
		4 × I	0.889	0.897	0.903	0.899	0.905	0.912	0.91	0.909	0.912	0.913	0.915	0.914	0.912	0.918	**0.919**	0.915
		5 × I	0.898	0.895	0.905	0.91	0.91	0.908	0.911	0.914	0.907	0.915	0.913	0.913	0.917	0.912	**0.919**	0.915
	9	1 × I	0.885	0.886	0.893	0.896	0.901	0.896	0.898	0.9	0.899	0.9	0.905	0.907	0.904	0.911	0.9	0.905
		2 × I	0.887	0.891	0.892	0.9	0.9	0.897	0.899	0.897	0.906	0.901	0.908	0.91	0.901	0.907	0.907	0.909
		3 × I	0.894	0.893	0.895	0.903	0.902	0.9	0.904	0.899	0.908	0.906	0.915	0.913	0.906	0.913	0.913	0.911
		4 × I	0.893	0.896	0.892	0.907	0.905	0.906	0.905	0.905	0.907	0.904	0.906	0.913	0.909	0.915	0.914	0.916
		5 × I	0.89	0.896	0.895	0.904	0.906	0.91	0.908	0.909	0.913	0.916	**0.917**	0.914	0.912	0.913	0.914	0.915
	11	1 × I	0.888	0.891	0.894	0.895	0.89	0.894	0.897	0.902	0.9	0.896	0.9	0.901	0.9	0.908	0.903	0.899
		2 × I	0.888	0.886	0.893	0.896	0.896	0.894	0.898	0.908	0.903	0.914	0.9	0.903	0.908	0.91	0.907	0.905
		3 × I	0.893	0.896	0.899	0.897	0.899	0.897	0.902	0.908	0.899	0.904	0.911	0.908	0.908	0.913	0.908	0.909
		4 × I	0.89	0.893	0.893	0.897	0.905	0.902	0.905	0.907	0.905	0.907	0.912	0.914	0.913	**0.917**	0.912	0.915
		5 × I	0.888	0.897	0.904	0.899	0.901	0.906	0.913	0.906	0.913	0.91	0.912	**0.917**	0.915	0.914	0.915	0.915

https://doi.org/10.1371/journal.pone.0311041.t015

Table 16. Results of classification accuracy *acc* for the proposed approach and the Sensorless data sets: Two hidden layers, the method to substitute values of missing attributes in local tables—One artificial objects generated based on one original object, MLP networks aggregation using sum of weights and various number of neurons in the hidden layer (1AO-2HL-SUM). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	Second HL	No. of neurons in the first hidden layer															
			0.25 × I	0.5 × I	0.75 × I	1 × I	1.5 × I	1.75 × I	2 × I	2.5 × I	2.75 × I	3 × I	3.5 × I	3.75 × I	4 × I	4.5 × I	4.75 × I	5 × I
Sensorless imbalanced	3	1 × I	0.9	0.91	0.92	0.912	0.935	0.942	0.937	0.942	0.941	0.943	0.94	0.937	0.931	0.932	0.942	0.938
		2 × I	0.91	0.923	0.939	0.941	0.939	0.941	0.943	0.945	0.941	0.936	0.941	0.943	0.941	0.941	0.938	0.946
		3 × I	0.937	0.931	0.939	0.937	0.939	0.945	**0.949**	0.939	0.938	0.945	0.94	0.939	0.937	0.939	0.935	0.937
		4 × I	0.923	0.942	0.932	0.945	0.947	0.947	0.942	0.943	0.945	0.937	0.939	0.937	0.942	0.94	0.938	0.934
		5 × I	0.924	0.94	0.944	0.939	0.942	0.942	0.946	0.943	0.948	0.946	0.945	0.941	0.933	0.934	0.936	0.934
	5	1 × I	0.9	0.912	0.919	0.92	0.923	0.93	0.932	0.927	0.926	0.928	0.928	0.93	0.93	0.925	0.924	0.924
		2 × I	0.915	0.921	0.926	0.917	0.931	0.927	0.931	**0.934**	0.932	0.931	0.931	0.926	0.926	0.927	0.927	0.927
		3 × I	0.916	0.919	0.927	0.923	0.928	0.925	0.932	0.93	0.925	0.928	0.931	0.921	0.922	0.927	0.926	0.927
		4 × I	0.916	0.918	0.924	0.929	0.926	0.929	0.931	0.923	0.931	0.927	0.924	0.926	0.928	0.925	0.927	0.917
		5 × I	0.915	0.926	0.93	0.928	0.923	0.931	0.927	0.925	0.927	0.925	0.924	0.916	0.921	0.923	0.923	0.918
	7	1 × I	0.906	0.922	0.918	0.926	0.927	0.932	0.936	0.935	0.936	0.933	0.93	0.933	0.936	0.934	0.933	0.933
		2 × I	0.904	0.931	0.927	0.932	0.933	0.934	0.933	0.934	0.938	0.938	0.932	0.933	0.931	0.935	0.929	0.936
		3 × I	0.92	0.924	0.928	0.931	0.939	0.928	0.939	0.938	0.929	0.936	0.936	0.934	0.93	0.928	0.93	0.93
		4 × I	0.922	0.933	0.931	0.937	0.939	0.93	0.939	0.934	0.933	0.935	0.93	0.935	0.931	0.929	0.934	0.929
		5 × I	0.922	0.93	0.936	0.933	**0.942**	0.937	0.927	0.933	0.932	0.936	0.933	0.931	0.93	0.933	0.927	0.933
	9	1 × I	0.91	0.913	0.923	0.922	0.921	0.931	0.929	0.934	0.93	0.932	0.932	0.935	0.93	0.932	0.93	0.93
		2 × I	0.909	0.923	0.923	0.936	0.933	0.934	0.936	0.935	0.932	0.932	0.936	0.93	0.932	0.935	0.935	0.937
		3 × I	0.917	0.931	0.928	0.934	0.938	0.936	0.934	0.937	0.929	0.934	0.934	0.934	0.936	0.931	0.93	0.936
		4 × I	0.927	0.932	0.932	0.931	0.933	0.934	**0.942**	0.93	0.937	0.929	0.928	0.934	0.935	0.936	0.93	0.934
		5 × I	0.92	0.93	0.93	0.932	0.933	0.934	0.934	0.928	0.935	0.937	0.935	0.932	0.926	0.935	0.924	0.932
	11	1 × I	0.908	0.911	0.923	0.923	0.928	0.93	0.926	0.933	0.929	0.934	0.935	0.936	0.929	0.928	0.926	0.931
		2 × I	0.904	0.917	0.924	0.934	0.928	0.935	0.933	0.94	0.932	0.932	0.937	0.933	0.937	0.936	0.937	0.931
		3 × I	0.908	0.931	0.927	0.933	**0.941**	0.933	0.929	0.933	0.936	0.939	0.93	0.932	0.935	0.934	0.933	0.937
		4 × I	0.92	0.927	0.932	0.935	0.936	0.936	0.936	0.936	0.936	0.929	0.94	0.937	0.938	0.935	0.935	0.934
		5 × I	0.918	0.929	0.931	0.935	0.934	0.938	0.937	0.932	0.935	0.933	0.931	0.938	0.928	0.93	0.93	0.931

https://doi.org/10.1371/journal.pone.0311041.t016

Table 17. Results of classification accuracy *acc* for the proposed approach and the Crowd Sourced data sets: Two hidden layers, the method to substitute values of missing attributes in local tables—One artificial objects generated based on one original object, MLP networks aggregation using sum of weights and various number of neurons in the hidden layer (1AO-2HL-SUM). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	Second HL	No. of neurons in the first hidden layer															
			0.25 × I	0.5 × I	0.75 × I	1 × I	1.5 × I	1.75 × I	2 × I	2.5 × I	2.75 × I	3 × I	3.5 × I	3.75 × I	4 × I	4.5 × I	4.75 × I	5 × I
Crowd Sourced imbalanced	3	1 × I	0.646	0.713	0.734	0.777	0.759	0.775	0.785	0.804	0.78	0.792	0.795	0.794	0.8	0.806	0.82	0.808
		2 × I	0.699	0.762	0.767	0.754	0.794	0.808	0.785	0.812	0.81	0.789	0.813	0.812	0.802	0.802	0.817	**0.838**
		3 × I	0.657	0.745	0.785	0.757	0.801	0.777	0.776	0.806	0.779	0.802	0.811	0.799	0.798	0.801	0.817	0.818
		4 × I	0.709	0.765	0.796	0.754	0.779	0.78	0.787	0.796	0.799	0.806	0.811	0.804	0.79	0.812	0.817	0.801
		5 × I	0.674	0.774	0.781	0.773	0.788	0.771	0.806	0.809	0.815	0.782	0.812	0.813	0.813	0.806	0.828	0.815
	5	1 × I	0.627	0.763	0.75	0.748	0.777	0.791	0.79	0.801	0.814	0.776	0.819	0.818	0.832	0.794	0.829	0.798
		2 × I	0.659	0.748	0.779	0.783	0.753	0.79	0.808	0.826	0.792	0.814	0.822	0.827	0.806	0.821	0.84	0.819
		3 × I	0.67	0.753	0.796	0.778	0.806	0.816	0.804	0.823	0.83	0.806	0.821	**0.845**	0.833	0.796	0.837	0.821
		4 × I	0.628	0.768	0.781	0.79	0.811	0.819	0.814	0.807	0.832	0.807	0.835	0.837	0.824	0.836	0.838	0.824
		5 × I	0.662	0.736	0.797	0.809	0.826	0.823	0.82	0.825	0.822	0.835	0.829	0.843	0.809	0.816	0.837	0.822
	7	1 × I	0.652	0.74	0.798	0.772	0.77	0.802	0.779	0.799	0.806	0.768	0.838	0.806	0.787	0.815	0.831	0.835
		2 × I	0.694	0.74	0.78	0.756	0.802	0.814	0.819	0.813	0.822	0.809	0.804	0.813	0.812	0.818	0.829	0.805
		3 × I	0.677	0.79	0.727	0.814	0.773	0.795	0.803	0.812	0.827	0.814	0.819	0.823	0.816	0.82	0.828	0.833
		4 × I	0.611	0.757	0.722	0.786	0.806	0.802	0.828	0.823	0.825	0.832	0.808	0.823	0.821	**0.842**	0.831	0.823
		5 × I	0.717	0.777	0.792	0.818	0.794	0.797	0.817	0.821	0.828	0.82	0.815	0.831	0.825	0.815	0.826	0.84
	9	1 × I	0.723	0.734	0.773	0.794	0.794	0.796	0.788	0.785	0.813	0.775	0.818	0.827	0.801	0.813	0.828	0.828
		2 × I	0.769	0.756	0.801	0.803	0.8	0.822	0.795	0.8	0.799	0.791	0.819	0.821	0.815	**0.839**	0.828	0.835
		3 × I	0.647	0.775	0.779	0.786	0.811	0.77	0.82	0.814	0.823	0.83	0.829	0.813	0.826	0.82	0.824	0.831
		4 × I	0.729	0.788	0.778	0.788	0.82	0.803	0.813	0.792	0.813	0.818	0.831	0.822	0.83	0.822	0.834	0.823
		5 × I	0.735	0.786	0.796	0.83	0.809	0.799	0.823	0.818	0.824	0.827	0.828	0.819	0.837	0.813	0.825	0.829
	11	1 × I	0.619	0.76	0.777	0.798	0.805	0.789	0.812	0.806	0.824	0.837	0.816	0.844	0.829	0.832	0.83	0.835
		2 × I	0.772	0.767	0.786	0.815	0.77	0.823	0.826	0.835	0.833	0.809	0.831	0.84	0.836	0.841	0.825	0.844
		3 × I	0.741	0.791	0.806	0.8	0.808	0.835	0.794	0.833	0.812	0.826	0.821	0.844	0.831	**0.861**	0.823	0.845
		4 × I	0.676	0.79	0.802	0.773	0.828	0.807	0.837	0.829	0.831	0.826	0.833	0.842	0.839	0.844	0.842	0.841
		5 × I	0.734	0.769	0.809	0.808	0.823	0.818	0.82	0.83	0.837	0.826	0.844	0.837	0.837	0.816	0.826	0.838
Crowd Sourced balanced	3	1 × I	0.463	0.676	0.705	0.769	0.759	0.778	0.807	0.815	0.825	0.82	0.82	0.832	0.828	0.852	0.812	0.786
		2 × I	0.453	0.688	0.741	0.78	0.763	0.792	0.785	0.835	0.832	0.823	0.841	0.829	0.842	0.856	0.863	0.847
		3 × I	0.481	0.652	0.761	0.781	0.802	0.806	0.821	0.816	0.825	0.826	0.866	0.834	0.854	0.871	**0.879**	0.863
		4 × I	0.579	0.661	0.756	0.744	0.805	0.802	0.849	0.84	0.847	0.838	0.84	0.852	0.834	0.846	0.846	0.868
		5 × I	0.6	0.698	0.684	0.725	0.809	0.84	0.807	0.836	0.839	0.826	0.856	0.848	0.837	0.868	0.846	0.873
	5	1 × I	0.436	0.703	0.673	0.751	0.756	0.742	0.746	0.823	0.773	0.86	0.869	0.858	0.811	0.868	0.862	0.835
		2 × I	0.521	0.688	0.771	0.763	0.82	0.785	0.84	0.838	0.838	0.832	0.873	0.858	0.854	0.875	0.864	0.866
		3 × I	0.618	0.751	0.731	0.817	0.821	0.808	0.823	0.838	0.86	0.87	0.871	0.882	0.869	0.863	0.859	0.872
		4 × I	0.636	0.708	0.739	0.783	0.817	0.838	0.849	0.836	0.848	0.867	0.872	0.876	0.887	0.885	0.875	**0.894**
		5 × I	0.6	0.743	0.793	0.787	0.833	0.869	0.856	0.872	0.885	0.867	0.87	0.883	0.859	0.884	0.894	0.89
	7	1 × I	0.472	0.709	0.701	0.705	0.76	0.743	0.798	0.739	0.823	0.741	0.837	0.79	0.835	0.871	0.87	0.821
		2 × I	0.639	0.718	0.77	0.785	0.801	0.791	0.741	0.849	0.842	0.858	0.882	0.866	0.845	0.889	0.875	0.876
		3 × I	0.566	0.713	0.75	0.8	0.77	0.816	0.82	0.796	0.854	0.835	0.837	0.879	0.859	0.853	0.885	0.887
		4 × I	0.612	0.748	0.792	0.78	0.824	0.808	0.834	0.85	0.855	0.867	0.858	0.867	0.878	0.891	0.884	**0.899**
		5 × I	0.622	0.75	0.776	0.806	0.849	0.837	0.843	0.854	0.869	0.867	0.865	0.881	0.875	0.887	0.881	0.886
	9	1 × I	0.62	0.702	0.724	0.723	0.779	0.797	0.751	0.83	0.82	0.816	0.832	0.854	0.86	0.858	0.853	0.864
		2 × I	0.58	0.675	0.698	0.731	0.817	0.76	0.83	0.83	0.829	0.854	0.856	0.869	0.851	0.874	0.885	0.871
		3 × I	0.495	0.705	0.789	0.758	0.823	0.825	0.809	0.857	0.853	0.846	0.853	0.865	0.877	0.873	0.859	0.892
		4 × I	0.662	0.686	0.793	0.789	0.838	0.833	0.825	0.84	0.867	0.843	0.873	0.845	0.877	0.873	0.88	0.883
		5 × I	0.608	0.753	0.782	0.811	0.827	0.855	0.818	0.849	0.869	0.872	0.878	0.887	0.881	0.874	0.873	**0.893**
	11	1 × I	0.637	0.675	0.7	0.749	0.729	0.753	0.719	0.83	0.81	0.82	0.831	0.807	0.846	0.83	0.842	0.858
		2 × I	0.654	0.736	0.751	0.749	0.761	0.808	0.78	0.803	0.825	0.859	0.852	0.84	0.875	0.874	0.873	0.885
		3 × I	0.679	0.734	0.762	0.803	0.786	0.848	0.787	0.853	0.859	0.852	0.868	0.854	0.884	0.895	0.89	0.885
		4 × I	0.659	0.738	0.802	0.805	0.825	0.817	0.831	0.819	0.857	0.869	0.876	0.89	0.885	0.882	0.897	0.878
		5 × I	0.737	0.757	0.774	0.787	0.843	0.849	0.871	0.874	0.873	0.853	0.868	0.87	0.884	0.892	**0.903**	0.899

https://doi.org/10.1371/journal.pone.0311041.t017

Table 18. Results of classification accuracy *acc* for the proposed approach and the Vehicle data sets: Two hidden layers, the method to substitute values of missing attributes in local tables—Three artificial objects generated based on one original object, MLP networks aggregation using sum of weights and various number of neurons in the hidden layer (3AO-2HL-SUM). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	Second HL	No. of neurons in the first hidden layer															
			0.25 × I	0.5 × I	0.75 × I	1 × I	1.5 × I	1.75 × I	2 × I	2.5 × I	2.75 × I	3 × I	3.5 × I	3.75 × I	4 × I	4.5 × I	4.75 × I	5 × I
Vehicle imbalanced	3	1 × I	0.274	0.546	0.591	0.584	0.647	0.636	0.672	0.631	0.61	0.644	0.61	0.636	0.395	0.63	0.615	0.665
		2 × I	0.461	0.587	0.6	0.631	0.663	0.605	0.635	0.629	0.664	0.65	**0.713**	0.636	0.656	0.66	0.644	0.671
		3 × I	0.448	0.644	0.669	0.677	0.681	0.684	0.689	0.64	0.672	0.69	0.676	0.682	0.665	0.684	0.673	0.686
		4 × I	0.491	0.638	0.655	0.648	0.681	0.669	0.699	0.669	0.701	0.697	0.684	0.681	0.673	0.654	0.68	0.698
		5 × I	0.479	0.605	0.644	0.642	0.672	0.682	0.676	0.681	0.673	0.694	0.677	0.689	0.668	0.693	0.677	0.694
	5	1 × I	0.417	0.554	0.63	0.654	0.65	0.639	0.622	0.671	0.643	0.652	0.661	0.663	0.642	0.636	0.682	0.669
		2 × I	0.283	0.622	0.644	0.65	0.684	0.622	0.664	0.654	0.656	0.6	0.677	0.681	0.698	0.703	0.689	0.672
		3 × I	0.283	0.618	0.64	0.692	0.68	0.672	0.701	0.646	0.689	0.68	0.703	0.668	0.703	0.688	0.718	0.68
		4 × I	0.453	0.58	0.652	0.69	0.69	0.638	0.694	0.693	0.661	0.651	0.717	0.69	0.686	0.694	0.697	0.667
		5 × I	0.333	0.64	0.665	0.65	0.673	0.702	0.684	0.686	**0.72**	0.693	0.714	0.698	0.709	0.682	0.696	0.717
	7	1 × I	0.466	0.56	0.656	0.648	0.675	0.654	0.672	0.546	0.664	0.668	0.665	0.669	0.639	0.655	0.598	0.665
		2 × I	0.383	0.644	0.664	0.68	0.689	0.647	0.701	0.724	0.706	0.669	0.696	0.685	0.702	0.68	0.664	0.684
		3 × I	0.631	0.665	0.693	0.669	0.689	0.667	0.655	0.68	0.677	0.673	0.689	0.688	0.698	0.68	0.709	0.707
		4 × I	0.52	0.594	0.651	0.69	0.681	0.69	0.703	0.714	0.702	0.711	0.702	0.722	0.709	0.702	0.682	**0.724**
		5 × I	0.377	0.622	0.669	0.665	0.706	0.702	0.694	0.685	0.69	0.69	0.698	0.694	0.693	0.71	0.702	0.699
	9	1 × I	0.606	0.416	0.629	0.671	0.66	0.675	0.588	0.591	0.639	0.654	0.626	0.644	0.668	0.667	0.664	0.655
		2 × I	0.524	0.659	0.63	0.644	0.663	0.693	0.65	0.672	0.678	0.684	0.709	0.677	0.64	0.681	0.675	0.676
		3 × I	0.276	0.541	0.668	0.667	0.706	0.694	0.681	0.681	0.671	0.677	0.678	0.676	0.703	0.667	0.681	0.677
		4 × I	0.4	0.512	0.657	0.682	0.693	0.697	0.682	0.675	0.68	0.69	0.685	0.682	0.663	0.698	0.68	0.69
		5 × I	0.5	0.629	0.686	0.68	0.693	**0.723**	0.699	0.673	0.673	0.673	0.697	0.685	0.692	0.686	0.692	0.707
	11	1 × I	0.283	0.652	0.686	0.665	0.664	0.675	0.671	0.663	0.694	0.665	0.684	0.689	0.44	0.671	0.676	0.69
		2 × I	0.566	0.672	0.615	0.689	0.655	0.706	0.702	0.693	0.693	0.69	0.682	0.685	0.714	0.696	0.684	0.703
		3 × I	0.52	0.684	0.692	0.675	0.689	0.68	0.668	0.719	0.698	0.696	0.718	0.689	0.694	0.696	0.69	0.713
		4 × I	0.633	0.625	0.698	0.667	0.689	0.684	0.693	0.71	0.71	0.69	0.707	0.707	0.702	0.677	0.703	0.694
		5 × I	0.35	0.68	**0.739**	0.667	0.709	0.71	0.701	0.705	0.696	0.707	0.707	0.723	0.697	0.663	0.72	0.718
Vehicle blanced	3	1 × I	0.567	0.625	0.566	0.698	0.636	0.685	0.612	0.685	0.651	0.677	0.664	0.667	0.684	0.702	0.644	0.283
		2 × I	0.672	0.572	0.651	0.71	0.701	0.65	0.66	0.668	0.723	0.669	0.706	0.724	0.706	0.685	0.698	0.677
		3 × I	0.677	0.512	0.711	0.72	0.707	0.681	0.692	0.723	0.722	0.723	0.68	0.715	0.706	0.709	0.697	0.709
		4 × I	0.379	0.623	0.676	0.705	0.694	0.713	0.715	0.714	0.731	0.701	**0.743**	0.715	0.694	0.719	0.689	0.724
		5 × I	0.669	0.469	0.72	0.69	0.707	0.722	0.684	0.706	0.707	0.676	0.699	0.684	0.701	0.693	0.727	0.722
	5	1 × I	0.356	0.675	0.606	0.689	0.648	0.686	0.707	0.702	0.678	0.677	0.713	0.664	0.686	0.613	0.667	0.686
		2 × I	0.382	0.681	0.714	0.714	0.665	0.694	0.702	0.684	0.719	0.719	0.72	0.714	0.727	0.693	0.719	0.728
		3 × I	0.567	0.689	0.69	0.66	0.705	0.727	0.69	0.707	0.696	0.686	0.72	0.707	0.715	0.722	0.714	0.702
		4 × I	0.49	0.673	0.61	0.699	0.718	0.698	0.696	0.713	0.727	0.705	0.713	**0.734**	0.722	0.694	0.722	0.713
		5 × I	0.631	0.705	0.685	0.692	0.72	0.706	0.689	0.73	0.709	0.727	0.717	0.706	0.724	0.73	0.722	0.714
	7	1 × I	0.509	0.65	0.707	0.702	0.677	0.707	0.705	0.727	0.747	0.437	0.719	0.706	0.719	0.283	0.728	0.719
		2 × I	0.487	0.685	0.689	0.681	0.724	0.701	0.717	0.686	0.719	0.72	0.744	0.72	0.732	0.722	0.741	0.743
		3 × I	0.281	0.686	0.705	0.702	0.693	0.728	0.726	0.719	0.731	0.749	**0.765**	0.732	0.741	0.74	0.71	0.73
		4 × I	0.433	0.572	0.672	0.697	0.723	0.732	0.727	0.728	0.707	0.739	0.736	0.756	0.734	0.743	0.723	0.709
		5 × I	0.28	0.72	0.735	0.731	0.714	0.717	0.744	0.736	0.744	0.743	0.744	0.744	0.722	0.732	0.73	0.739
	9	1 × I	0.283	0.678	0.651	0.681	0.661	0.663	0.682	0.701	0.672	0.718	0.731	0.705	0.696	0.626	0.69	0.696
		2 × I	0.526	0.638	0.65	0.642	0.701	0.693	0.681	0.688	0.706	0.689	0.713	0.689	0.705	0.724	0.701	0.705
		3 × I	0.283	0.625	0.669	0.64	0.693	0.703	0.711	0.694	0.722	0.702	0.718	0.707	0.724	0.728	0.686	0.707
		4 × I	0.605	0.672	0.678	0.707	0.689	0.717	0.722	0.696	0.701	0.705	0.715	0.713	0.736	0.715	0.717	0.701
		5 × I	0.57	0.731	0.696	0.709	0.697	0.722	0.703	0.727	0.73	0.717	0.71	0.701	0.717	0.722	**0.741**	0.705
	11	1 × I	0.579	0.648	0.661	0.655	0.682	0.696	0.706	0.705	0.702	0.711	0.701	0.702	0.709	0.698	0.707	0.69
		2 × I	0.497	0.642	0.709	0.711	0.719	0.713	0.714	0.684	0.703	0.724	0.73	0.706	0.724	0.728	0.711	0.728
		3 × I	0.648	0.631	0.714	0.719	0.734	0.699	0.72	0.732	0.711	0.723	0.705	0.735	0.714	0.714	0.696	0.734
		4 × I	0.503	0.673	0.715	0.706	0.743	0.744	**0.747**	0.706	0.703	0.726	0.706	0.715	0.722	0.701	0.718	0.723
		5 × I	0.644	0.597	0.718	0.698	0.713	0.726	0.736	0.714	0.713	0.727	0.726	0.732	0.727	0.732	0.724	0.715

https://doi.org/10.1371/journal.pone.0311041.t018

Table 19. Results of classification accuracy *acc* for the proposed approach and the Dry Bean data sets: Two hidden layers, the method to substitute values of missing attributes in local tables—Three artificial objects generated based on one original object, MLP networks aggregation using sum of weights and various number of neurons in the hidden layer (3AO-2HL-SUM). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	Second HL	No. of neurons in the first hidden layer															
			0.25 × I	**0.5 × I**	**0.75 × I**	**1 × I**	**1.5 × I**	**1.75 × I**	**2 × I**	**2.5 × I**	**2.75 × I**	**3 × I**	**3.5 × I**	**3.75 × I**	**4 × I**	**4.5 × I**	**4.75 × I**	**5 × I**
Dry Bean imbalanced	3	1 × I	0.887	0.893	0.89	0.902	0.903	0.91	0.903	0.9	0.903	0.909	0.906	0.914	0.915	0.915	0.914	0.913
		2 × I	0.898	0.898	0.905	0.906	0.911	0.91	0.909	0.908	0.908	0.912	0.916	0.91	0.915	0.911	0.914	0.917
		3 × I	0.892	0.899	0.913	0.9	0.91	0.91	0.914	0.914	0.913	0.914	0.915	0.917	0.916	0.916	**0.919**	0.916
		4 × I	0.889	0.907	0.908	0.912	0.91	0.909	0.914	0.915	0.914	0.916	0.918	0.916	0.917	0.915	**0.919**	0.918
		5 × I	0.901	0.907	0.898	0.912	0.918	0.913	0.917	0.917	0.918	0.917	0.917	0.917	**0.919**	0.918	0.917	0.918
	5	1 × I	0.888	0.885	0.892	0.897	0.901	0.91	0.908	0.905	0.902	0.907	0.904	0.909	0.906	0.902	0.91	0.901
		2 × I	0.895	0.897	0.899	0.901	0.904	0.904	0.906	0.908	0.914	0.913	0.907	0.909	0.912	0.913	0.914	0.907
		3 × I	0.896	0.9	0.909	0.911	0.914	0.905	0.914	0.914	0.909	0.913	0.916	0.916	0.914	0.915	0.915	0.916
		4 × I	0.89	0.9	0.902	0.903	0.909	0.906	0.909	0.914	0.914	**0.917**	0.913	0.915	0.914	0.916	0.915	0.915
		5 × I	0.888	0.898	0.909	0.913	0.901	0.908	0.91	0.909	**0.917**	0.915	0.916	0.916	0.913	0.915	0.914	0.913
	7	1 × I	0.885	0.89	0.892	0.893	0.899	0.895	0.9	0.904	0.904	0.901	0.904	0.9	0.899	0.9	0.904	0.907
		2 × I	0.884	0.896	0.893	0.895	0.902	0.903	0.897	0.904	0.901	0.907	0.9	0.905	0.907	0.91	0.909	0.906
		3 × I	0.888	0.895	0.894	0.898	0.9	0.899	0.898	0.907	0.91	0.909	0.91	0.906	0.914	0.911	0.908	0.91
		4 × I	0.893	0.894	0.901	0.902	0.906	0.911	0.91	0.902	0.908	0.907	0.906	0.909	0.912	0.912	0.913	0.914
		5 × I	0.884	0.895	0.906	0.909	0.908	0.911	0.905	0.909	0.91	0.91	0.912	0.91	0.91	0.914	**0.915**	0.912
	9	1 × I	0.886	0.886	0.894	0.89	0.891	0.896	0.893	0.901	0.897	0.895	0.899	0.894	0.903	0.9	0.903	0.903
		2 × I	0.883	0.888	0.896	0.891	0.9	0.901	0.897	0.903	0.9	0.902	0.903	0.906	0.909	0.902	0.903	0.907
		3 × I	0.895	0.892	0.892	0.899	0.899	0.905	0.899	0.908	0.907	0.903	0.909	0.907	0.908	0.904	0.9	0.91
		4 × I	0.893	0.896	0.893	0.898	0.899	0.906	0.906	0.902	0.908	0.909	0.908	0.909	0.906	0.907	0.914	0.912
		5 × I	0.89	0.895	0.896	0.901	0.904	0.9	0.911	0.909	0.91	0.914	0.907	**0.915**	0.91	**0.915**	0.912	0.911
	11	1 × I	0.881	0.889	0.89	0.896	0.894	0.893	0.896	0.897	0.896	0.895	0.895	0.897	0.902	0.896	0.897	0.899
		2 × I	0.888	0.893	0.896	0.897	0.894	0.893	0.897	0.9	0.902	0.901	0.9	0.901	0.901	0.904	0.902	0.905
		3 × I	0.889	0.894	0.894	0.893	0.897	0.9	0.902	0.902	0.9	0.901	0.906	0.904	0.904	0.898	0.903	0.903
		4 × I	0.889	0.893	0.896	0.899	0.897	0.9	0.898	0.906	0.905	0.905	0.904	0.904	0.905	0.906	0.902	0.908
		5 × I	0.895	0.894	0.895	0.901	0.9	0.901	0.9	0.903	0.904	0.907	0.902	0.906	0.902	**0.908**	0.906	**0.908**
Dry Bean blanced	3	1 × I	0.89	0.896	0.902	0.91	0.903	0.908	0.908	0.909	0.913	0.912	0.91	0.914	0.908	0.914	0.91	0.914
		2 × I	0.902	0.898	0.903	0.908	0.906	0.912	0.914	0.912	0.915	0.915	0.915	0.912	0.918	0.918	0.916	0.919
		3 × I	0.91	0.904	0.903	0.902	0.912	0.916	0.913	0.912	0.913	0.905	0.917	0.913	0.905	0.917	0.917	0.917
		4 × I	0.905	0.903	0.906	0.913	0.913	0.913	0.914	0.916	0.918	0.916	0.915	0.919	0.915	0.916	0.917	0.916
		5 × I	0.87	0.894	0.908	0.915	0.91	0.913	0.915	0.919	0.918	0.916	0.919	0.917	0.918	0.917	**0.92**	0.915
	5	1 × I	0.888	0.893	0.895	0.893	0.896	0.908	0.905	0.909	0.912	0.914	0.902	0.912	0.915	0.916	0.914	0.915
		2 × I	0.907	0.907	0.904	0.909	0.905	0.905	0.911	0.91	0.914	0.913	0.917	0.916	0.915	0.916	0.917	0.917
		3 × I	0.891	0.909	0.91	0.91	0.91	0.914	0.913	0.918	0.916	0.916	0.915	0.918	0.916	0.915	0.916	0.917
		4 × I	0.887	0.909	0.907	0.907	0.911	0.917	0.917	0.915	0.912	0.915	0.918	0.914	0.918	0.915	**0.919**	0.916
		5 × I	0.893	0.91	0.905	0.914	0.915	0.914	0.916	0.917	0.918	**0.919**	0.916	0.917	0.918	0.917	0.918	0.918
	7	1 × I	0.883	0.891	0.893	0.897	0.896	0.897	0.903	0.898	0.903	0.905	0.902	0.902	0.901	0.906	0.908	0.908
		2 × I	0.895	0.895	0.896	0.898	0.906	0.901	0.908	0.906	0.915	0.912	0.909	0.911	0.902	0.911	0.907	0.911
		3 × I	0.898	0.901	0.903	0.911	0.905	0.903	0.909	0.906	0.91	0.908	0.913	0.913	0.912	0.911	0.916	0.916
		4 × I	0.889	0.901	0.904	0.899	0.902	0.911	0.91	0.908	0.913	0.918	0.913	0.912	0.916	0.916	0.917	**0.918**
		5 × I	0.899	0.897	0.898	0.902	0.904	0.908	0.906	0.915	0.912	0.916	0.916	0.912	0.915	0.917	0.916	**0.918**
	9	1 × I	0.884	0.89	0.891	0.89	0.901	0.896	0.897	0.904	0.9	0.897	0.902	0.902	0.909	0.898	0.907	0.903
		2 × I	0.892	0.888	0.891	0.893	0.901	0.901	0.902	0.898	0.907	0.903	0.908	0.901	0.91	0.912	0.909	0.911
		3 × I	0.888	0.891	0.899	0.896	0.908	0.906	0.905	0.904	0.91	0.91	0.908	0.911	0.907	0.915	0.913	0.913
		4 × I	0.888	0.895	0.893	0.908	0.905	0.901	0.901	0.908	0.905	0.911	0.911	0.912	0.908	0.911	0.915	0.913
		5 × I	0.891	0.896	0.901	0.906	0.91	0.907	0.903	0.909	0.909	0.909	0.914	**0.916**	0.913	0.913	0.913	0.912
	11	1 × I	0.882	0.888	0.891	0.897	0.895	0.893	0.895	0.898	0.897	0.903	0.902	0.898	0.902	0.905	0.898	0.902
		2 × I	0.884	0.892	0.897	0.895	0.901	0.9	0.898	0.903	0.903	0.9	0.905	0.903	0.904	0.903	0.903	0.911
		3 × I	0.891	0.894	0.894	0.896	0.896	0.901	0.902	0.91	0.905	0.904	0.904	0.909	0.91	0.913	0.912	0.909
		4 × I	0.887	0.891	0.897	0.899	0.9	0.903	0.908	0.908	0.912	0.906	0.91	0.912	0.91	0.914	0.914	0.91
		5 × I	0.89	0.899	0.898	0.901	0.908	0.905	0.906	0.909	0.913	0.909	0.907	0.906	0.916	0.915	**0.92**	0.917

https://doi.org/10.1371/journal.pone.0311041.t019

Table 20. Results of classification accuracy *acc* for the proposed approach and the Sensorless data sets: Two hidden layers, the method to substitute values of missing attributes in local tables—Three artificial objects generated based on one original object, MLP networks aggregation using sum of weights and various number of neurons in the hidden layer (3AO-2HL-SUM). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	Second HL	No. of neurons in the first hidden layer															
			0.25 × I	**0.5 × I**	**0.75 × I**	**1 × I**	**1.5 × I**	**1.75 × I**	**2 × I**	**2.5 × I**	**2.75 × I**	**3 × I**	**3.5 × I**	**3.75 × I**	**4 × I**	**4.5 × I**	**4.75 × I**	**5 × I**
Sensorless imbalanced	3	1 × I	0.917	0.92	0.928	0.929	0.948	0.94	0.938	0.945	0.942	0.943	0.935	0.942	0.942	0.937	0.94	0.941
		2 × I	0.917	0.933	0.934	0.944	0.948	0.942	0.939	0.943	0.942	0.945	0.942	0.945	0.942	0.942	0.94	0.943
		3 × I	0.937	0.931	0.945	0.945	**0.952**	0.944	0.946	0.943	0.947	0.945	0.942	0.938	0.941	0.934	0.942	0.947
		4 × I	0.927	0.946	0.942	0.949	0.944	0.945	0.941	0.947	0.942	0.945	0.941	0.944	0.943	0.939	0.93	0.942
		5 × I	0.929	0.949	0.943	0.952	0.947	0.946	0.949	0.945	0.946	0.95	0.943	0.94	0.945	0.937	0.943	0.943
	5	1 × I	0.902	0.911	0.916	0.921	0.918	0.925	0.923	0.919	0.92	0.927	0.925	0.919	0.919	0.918	0.913	0.925
		2 × I	0.904	0.912	0.924	0.928	0.926	0.928	0.924	0.923	0.923	0.926	0.926	0.918	0.918	0.92	0.922	0.92
		3 × I	0.911	0.915	0.919	0.922	0.924	0.922	0.927	0.927	0.915	0.923	0.916	0.922	0.92	0.916	0.928	0.909
		4 × I	0.907	0.914	**0.932**	0.922	0.919	0.928	0.924	0.922	0.919	0.916	0.913	0.914	0.919	0.914	0.916	0.923
		5 × I	0.917	0.915	0.919	0.924	0.919	0.916	0.926	0.913	0.917	0.915	0.914	0.922	0.915	0.92	0.912	0.918
	7	1 × I	0.905	0.915	0.912	0.926	0.921	0.927	0.924	0.93	0.93	0.933	0.93	0.924	0.927	0.93	0.928	0.929
		2 × I	0.916	0.92	0.92	0.923	0.917	0.923	0.93	0.93	0.929	0.928	0.918	0.929	0.927	0.93	0.93	0.93
		3 × I	0.915	0.918	0.927	0.928	0.929	0.93	0.924	0.921	0.922	0.931	0.929	0.932	0.933	0.92	0.927	0.929
		4 × I	0.918	0.918	0.927	0.929	0.929	0.927	0.925	0.93	0.93	0.926	0.928	0.929	**0.934**	0.926	0.92	0.93
		5 × I	0.92	0.921	0.924	0.925	0.929	0.93	0.923	0.922	0.927	0.929	0.931	0.922	0.926	0.924	0.925	0.922
	9	1 × I	0.896	0.912	0.921	0.918	0.927	0.926	0.926	0.924	0.925	0.932	0.925	0.935	0.93	0.926	0.931	0.925
		2 × I	0.899	0.923	0.925	0.923	0.928	**0.937**	0.935	0.928	0.925	0.935	0.926	0.923	0.931	0.926	0.923	0.93
		3 × I	0.909	0.914	0.927	0.932	0.929	0.934	0.926	0.924	0.929	0.922	0.931	0.934	0.931	0.926	0.925	0.927
		4 × I	0.911	0.922	0.928	0.927	0.924	0.927	0.931	0.928	0.933	0.93	0.931	0.928	0.931	0.923	0.926	0.928
		5 × I	0.911	0.926	0.925	0.921	0.928	0.928	0.93	0.926	0.921	0.919	0.919	0.922	0.926	0.928	0.929	0.926
	11	1 × I	0.879	0.908	0.914	0.925	0.925	0.931	0.93	0.932	0.921	0.93	0.932	0.929	0.934	0.933	0.927	0.922
		2 × I	0.894	0.916	0.921	0.918	0.93	0.926	0.926	0.932	0.924	0.93	0.929	0.931	0.93	0.932	0.932	0.932
		3 × I	0.908	0.917	0.92	**0.935**	0.928	0.934	0.934	0.93	0.929	0.932	0.933	0.931	0.929	0.93	0.929	0.931
		4 × I	0.898	0.91	0.929	0.932	0.932	0.934	0.929	0.929	0.932	0.929	0.932	0.925	0.927	0.923	0.93	0.926
		5 × I	0.913	0.915	0.929	0.926	0.93	0.926	0.927	0.93	0.923	0.919	0.929	0.926	0.927	0.928	0.93	0.925

https://doi.org/10.1371/journal.pone.0311041.t020

Table 21. Results of classification accuracy *acc* for the proposed approach and the Crowd Sourced data sets: Two hidden layers, the method to substitute values of missing attributes in local tables—Three artificial objects generated based on one original object, MLP networks aggregation using sum of weights and various number of neurons in the hidden layer (3AO-2HL-SUM). Designation I is used for the number of neurons in the input layer.

Data set	No. tables	Second HL	No. of neurons in the first hidden layer															
			0.25 × I	0.5 × I	0.75 × I	1 × I	1.5 × I	1.75 × I	2 × I	2.5 × I	2.75 × I	3 × I	3.5 × I	3.75 × I	4 × I	4.5 × I	4.75 × I	5 × I
Crowd Sourced imbalanced	3	1 × I	0.676	0.738	0.742	0.728	0.755	0.779	0.759	0.77	0.752	0.774	0.764	0.778	0.767	0.787	0.789	0.768
		2 × I	0.665	0.747	0.744	0.764	0.776	0.776	0.756	0.752	0.772	0.777	0.803	0.79	0.788	0.784	0.787	0.774
		3 × I	0.691	0.753	0.775	0.735	0.77	0.77	0.768	0.792	0.775	0.791	0.78	0.784	0.788	0.786	0.794	0.805
		4 × I	0.643	0.729	0.74	0.762	0.759	0.798	0.781	0.783	0.794	0.789	0.794	0.804	0.8	0.782	0.801	0.773
		5 × I	0.715	0.736	0.738	0.755	0.778	0.776	0.76	0.763	0.771	0.778	0.794	0.782	0.797	0.775	**0.806**	0.806
	5	1 × I	0.615	0.758	0.738	0.757	0.773	0.77	0.779	0.756	0.814	0.782	0.805	0.799	0.792	0.788	0.785	0.835
		2 × I	0.68	0.734	0.718	0.773	0.772	0.777	0.757	0.769	0.773	0.794	0.826	0.823	0.774	0.814	0.809	0.833
		3 × I	0.702	0.755	0.748	0.782	0.778	0.794	0.776	0.793	0.807	0.812	0.812	0.81	0.796	0.815	0.821	**0.838**
		4 × I	0.691	0.74	0.766	0.771	0.782	0.784	0.794	0.808	0.777	0.815	0.816	0.807	0.8	0.816	0.826	0.805
		5 × I	0.692	0.739	0.761	0.795	0.776	0.799	0.798	0.792	0.807	0.807	0.819	0.814	0.818	0.822	0.832	0.829
	7	1 × I	0.677	0.751	0.72	0.759	0.761	0.759	0.776	0.785	0.781	0.785	0.786	0.773	0.785	0.788	0.812	0.794
		2 × I	0.694	0.761	0.781	0.746	0.769	0.793	0.794	0.8	0.797	0.801	0.798	0.82	0.794	0.8	0.795	0.821
		3 × I	0.712	0.775	0.775	0.756	0.784	0.777	0.797	0.79	0.79	0.8	0.809	0.812	0.814	0.822	0.813	0.831
		4 × I	0.614	0.76	0.741	0.776	0.793	0.792	0.788	0.799	0.77	0.817	0.827	0.82	0.818	0.823	0.826	0.83
		5 × I	0.717	0.76	0.757	0.753	0.803	0.784	0.813	0.801	0.811	0.794	0.825	0.803	0.82	**0.834**	0.827	0.824
	9	1 × I	0.756	0.77	0.775	0.763	0.785	0.788	0.738	0.785	0.809	0.786	0.806	0.804	0.816	0.812	0.827	0.819
		2 × I	0.74	0.765	0.783	0.776	0.811	0.785	0.817	0.825	0.804	0.806	0.824	0.813	0.814	0.837	0.794	0.829
		3 × I	0.675	0.779	0.754	0.758	0.819	0.815	0.807	0.8	0.804	0.825	0.815	0.812	0.832	0.829	0.825	0.827
		4 × I	0.722	0.793	0.816	0.774	0.782	0.811	0.791	0.824	0.82	0.829	0.81	**0.844**	0.817	0.832	0.828	0.83
		5 × I	0.719	0.773	0.79	0.782	0.835	0.793	0.811	0.799	0.812	0.829	0.829	0.821	**0.838**	0.835	0.843	0.842
	11	1 × I	0.706	0.767	0.749	0.776	0.773	0.741	0.765	0.812	0.802	0.813	0.791	0.815	0.821	0.808	0.817	0.799
		2 × I	0.727	0.747	0.772	0.771	0.765	0.798	0.798	0.802	0.811	0.812	0.821	0.81	0.802	0.837	**0.814**	0.811
		3 × I	0.67	0.76	0.799	0.803	0.766	0.802	0.775	0.806	0.814	0.82	0.821	0.793	0.802	0.815	0.827	0.824
		4 × I	0.741	0.796	0.786	0.743	0.81	0.78	0.819	0.816	0.824	0.789	0.825	0.821	0.827	0.832	0.83	0.824
		5 × I	0.71	0.759	0.81	0.799	0.812	0.82	0.783	0.78	0.805	0.822	0.817	0.821	0.838	0.838	0.829	0.826
Crowd Sourced balanced	3	1 × I	0.57	0.68	0.715	0.711	0.736	0.703	0.703	0.737	0.736	0.762	0.739	0.744	0.74	0.753	0.761	0.758
		2 × I	0.63	0.653	0.715	0.697	0.759	0.727	0.755	0.728	0.724	0.725	0.739	0.765	0.806	0.758	0.814	0.774
		3 × I	0.596	0.669	0.712	0.69	0.649	0.784	0.731	0.763	0.727	0.784	0.734	0.791	0.782	0.785	0.721	0.77
		4 × I	0.591	0.647	0.702	0.666	0.711	0.676	0.755	0.728	0.753	0.763	0.737	0.803	0.748	0.763	0.759	0.786
		5 × I	0.575	0.704	0.734	0.708	0.738	0.72	0.746	0.769	0.731	0.745	0.692	0.775	0.781	0.763	0.774	0.789
	5	1 × I	0.442	0.68	0.701	0.734	0.787	0.792	0.761	0.827	0.813	0.811	0.795	0.848	0.83	0.827	0.847	0.861
		2 × I	0.386	0.713	0.732	0.746	0.768	0.8	0.803	0.83	0.816	0.829	0.823	0.831	0.848	0.861	0.847	0.857
		3 × I	0.528	0.612	0.695	0.774	0.776	0.809	0.806	0.832	0.825	0.819	0.839	0.83	0.849	0.864	**0.875**	0.811
		4 × I	0.466	0.737	0.744	0.763	0.791	0.812	0.803	0.838	0.816	0.814	0.84	0.861	0.84	0.843	0.866	0.872
		5 × I	0.521	0.727	0.717	0.794	0.835	0.791	0.817	0.792	0.854	0.822	0.826	0.821	0.834	0.859	0.833	0.872
	7	1 × I	0.403	0.716	0.75	0.745	0.78	0.775	0.725	0.832	0.833	0.84	0.863	0.831	0.854	0.85	0.845	0.866
		2 × I	0.585	0.649	0.762	0.765	0.81	0.805	0.796	0.837	0.849	0.848	0.863	0.857	0.852	0.874	0.857	0.872
		3 × I	0.394	0.683	0.721	0.762	0.827	0.82	0.813	0.862	0.854	0.844	0.876	0.844	0.852	0.864	0.877	0.891
		4 × I	0.513	0.739	0.735	0.794	0.814	0.812	0.83	0.852	0.849	0.853	0.844	0.869	0.852	0.878	0.879	0.883
		5 × I	0.6	0.726	0.788	0.781	0.796	0.842	0.849	0.864	0.864	0.846	0.864	0.868	0.873	0.875	**0.894**	0.879
	9	1 × I	0.541	0.717	0.747	0.775	0.777	0.799	0.805	0.794	0.801	0.815	0.849	0.839	0.879	0.862	0.853	0.849
		2 × I	0.547	0.699	0.738	0.733	0.808	0.801	0.837	0.838	0.843	0.83	0.833	0.841	0.858	0.879	0.875	0.877
		3 × I	0.551	0.679	0.795	0.765	0.789	0.815	0.826	0.858	0.831	0.847	0.87	0.867	0.87	0.885	0.873	0.883
		4 × I	0.333	0.704	0.747	0.766	0.837	0.796	0.824	0.836	0.853	0.862	0.878	0.855	0.874	0.883	0.888	0.886
		5 × I	0.486	0.726	0.751	0.776	0.792	0.849	0.858	0.848	0.864	0.873	0.883	0.889	0.864	0.879	0.842	**0.902**
	11	1 × I	0.475	0.664	0.703	0.739	0.786	0.818	0.787	0.815	0.818	0.829	0.841	0.837	0.857	0.809	0.858	0.875
		2 × I	0.522	0.703	0.72	0.779	0.738	0.806	0.817	0.836	0.845	0.851	0.863	0.855	0.866	0.876	0.893	0.894
		3 × I	0.662	0.689	0.749	0.774	0.79	0.794	0.784	0.845	0.874	0.865	0.865	0.874	0.84	0.877	0.872	0.894
		4 × I	0.594	0.73	0.769	0.765	0.821	0.84	0.838	0.859	0.858	0.868	0.872	0.853	0.888	0.881	0.893	0.888
		5 × I	0.607	0.737	0.792	0.762	0.816	0.825	0.833	0.873	0.854	0.871	0.875	0.887	0.881	0.887	0.896	**0.904**

https://doi.org/10.1371/journal.pone.0311041.t021

Table 22. Comparison of classification accuracy *acc* obtained for different numbers of artificial objects.

Data set	No. tables	1AO 1HL	3AO 1HL	1AO 2HL	3AO 2HL	
Vehicle imbalanced	3	0.693	**0.718**	**0.739**	0.732	AVG
	5	**0.703**	0.693	0.709	**0.711**	
	7	**0.69**	0.686	**0.724**	0.702	
	9	0.677	**0.694**	**0.713**	0.686	
	11	**0.696**	0.69	**0.706**	0.698	
Vehicle balanced	3	**0.727**	0.722	0.736	**0.76**	
	5	0.703	**0.72**	0.728	**0.732**	
	7	0.722	**0.732**	**0.757**	0.74	
	9	**0.73**	0.702	0.719	**0.728**	
	11	0.688	**0.692**	**0.723**	0.71	
Dry Bean imbalanced	3	**0.909**	**0.909**	0.906	**0.91**	
	5	0.908	**0.91**	**0.902**	0.901	
	7	**0.908**	0.907	**0.903**	0.902	
	9	**0.906**	**0.906**	**0.904**	0.903	
	11	0.906	**0.907**	**0.903**	**0.903**	
Dry Bean balanced	3	**0.913**	0.911	0.912	**0.918**	
	5	0.909	**0.91**	**0.901**	**0.901**	
	7	**0.907**	**0.907**	**0.903**	0.902	
	9	**0.909**	0.906	0.903	**0.904**	
	11	0.904	**0.905**	0.901	**0.904**	
Sensorless	3	**0.921**	0.92	0.95	**0.951**	
	5	**0.916**	0.914	**0.945**	0.944	
	7	**0.919**	**0.919**	**0.947**	0.943	
	9	**0.918**	0.916	**0.943**	0.94	
	11	0.918	**0.922**	**0.942**	0.939	
Crowd Sourced imbalanced	3	**0.827**	0.806	**0.838**	0.833	
	5	**0.84**	0.827	**0.875**	0.869	
	7	**0.851**	0.837	0.855	**0.867**	
	9	**0.851**	0.846	0.859	0.859	
	11	0.846	**0.871**	**0.876**	0.861	
Crowd Sourced balanced	3	**0.892**	0.833	**0.915**	0.858	
	5	**0.914**	0.892	**0.927**	0.919	
	7	**0.922**	0.916	0.918	0.918	
	9	0.916	**0.921**	0.915	**0.916**	
	11	0.904	**0.914**	0.921	**0.923**	

(*Continued*)

Table 22. (Continued)

Data set	No. tables	1AO 1HL	3AO 1HL	1AO 2HL	3AO 2HL	
Vehicle imbalanced	3	**0.675**	0.665	**0.713**	**0.713**	SUM
	5	**0.673**	0.671	0.715	**0.72**	
	7	0.682	**0.696**	**0.735**	0.724	
	9	0.697	**0.717**	**0.752**	0.723	
	11	**0.694**	0.688	**0.743**	0.739	
Vehicle balanced	3	**0.703**	0.677	0.726	**0.743**	
	5	0.71	**0.717**	**0.748**	0.734	
	7	0.723	**0.735**	0.759	**0.765**	
	9	**0.743**	0.711	**0.745**	0.741	
	11	**0.714**	0.705	**0.768**	0.747	
Dry Bean imbalanced	3	**0.912**	0.911	**0.92**	0.919	
	5	0.912	**0.913**	**0.918**	0.917	
	7	**0.914**	0.913	**0.917**	0.915	
	9	**0.914**	0.913	**0.915**	**0.915**	
	11	**0.913**	0.911	**0.914**	0.908	
Dry Bean balanced	3	**0.912**	0.911	0.919	**0.92**	
	5	**0.916**	0.913	0.917	**0.919**	
	7	0.912	**0.913**	**0.919**	0.918	
	9	**0.913**	0.912	**0.917**	0.916	
	11	**0.912**	**0.912**	0.917	**0.92**	
Sensorless	3	**0.92**	0.918	0.949	**0.952**	
	5	**0.911**	0.904	**0.934**	0.932	
	7	**0.915**	0.908	**0.942**	0.934	
	9	**0.917**	0.908	**0.942**	0.937	
	11	**0.914**	0.91	**0.941**	0.935	
Crowd Sourced imbalanced	3	0.796	**0.806**	**0.838**	0.806	
	5	**0.817**	0.802	**0.845**	0.838	
	7	**0.829**	0.818	**0.842**	0.834	
	9	**0.832**	0.808	0.839	**0.844**	
	11	0.827	**0.842**	**0.861**	0.838	
Crowd Sourced balanced	3	**0.864**	0.782	**0.879**	0.814	
	5	**0.874**	0.839	**0.894**	0.875	
	7	0.864	**0.866**	**0.899**	0.894	
	9	**0.86**	0.855	0.893	**0.902**	
	11	0.861	**0.863**	0.903	**0.904**	

https://doi.org/10.1371/journal.pone.0311041.t022

are marked in bold. Comparisons of experimental results with respect to different factors are made in separate sections.

5.1 Comparison of classification quality for different number of artificial objects created based on one original object in local table

Table 22 shows a comparison of the classification accuracy obtained for one and three generated artificial objects at various other settings (these are the best results which have been presented in bold in previous tables). For each setting and data set, the better result is marked in bold. In ninety-three cases, better results are obtained with one artificial object, and in fifty-four cases with three artificial objects generated. Thus, in most cases, generating just one

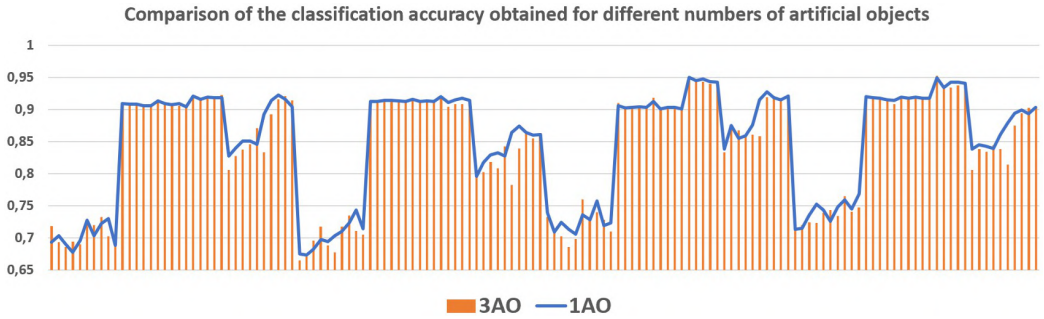


Fig 5. Comparison of classification accuracy *acc* obtained for different aggregation method.

https://doi.org/10.1371/journal.pone.0311041.g005

artificial object based on original object is enough to get better result. For statistical test [51], the two dependent groups are created (1AO and 3AO), each with one hundred and forty objects. Initially, the null hypothesis, H_θ is defined, whereby H_θ means there is no significance in terms of the number of artificial objects used in the model. The Wilcoxon test for dependent samples confirmed the statistical significance of the differences with $p = 0.0003$ (we have justification for rejecting the null hypothesis) and the medians are equal 0.9 and 0.893 for groups 1AO and 3AO respectively. This proves that actually using one artificial object on average generates better results than using three artificial objects.

In addition, Fig 5 is created on which the results for all the settings and data sets analyzed are marked with respect to different artificial objects generated (individual cases are not labeled on the x-axis for clarity). In the graph, one can observe that indeed using one artificial object in many cases generates better results.

5.2 Comparison of classification quality for different approaches: Average of weights and sum of weights to aggregating local neural networks

Table 23 shows a comparison of the classification accuracy obtained for methods for aggregating local networks: average and sum. For each setting and data set, the better result is marked in bold. In seventy-four cases, better results are obtained for average, and in sixty-nine cases for sum method. Thus, in most cases, the global network obtained by the sum of the weights provides better results. A statistical test is performed to confirm the significance of differences. The null hypothesis, H_θ is defined, whereby H_θ means there is no significance in terms of the method for aggregating local neural networks in the model. The two dependent groups are created (AVG and SUM), each with one hundred and forty objects. The Wilcoxon test for dependent samples confirmed the statistical significance of the differences in accuracy with $p = 0.04$ —so we have justification for rejecting the null hypothesis. The medians are equal 0.902 and 0.87 for groups AVG and SUM respectively. This proves that actually using the average approach generates better results. However, it should be noted that the results are very dependent on the data set. For the Vehicle and the Dry Bean data sets, it is definitely apparent that the sum method provides better results. However, for the Sensorless and the Crowd Sourced data sets, it is the average method that provides better results.

In addition, Fig 6 is created on which the results for all the settings and data sets analyzed are marked (individual cases are not labeled on the x-axis for clarity). In the graph, one can observer that indeed the average in many cases generates better results.

Table 23. Comparison of classification accuracy *acc* obtained for different aggregation method and different numbers of hidden layers.

Data set	No. tables	AVG 1HL	SUM 1HL	AVG 2HL	SUM 2HL	
Vehicle imbalanced	3	**0.693**	0.675	0.739	0.713	1AO
	5	0.703	**0.673**	0.709	0.715	
	7	**0.69**	0.682	0.724	0.735	
	9	0.677	**0.697**	0.713	0.752	
	11	**0.696**	0.694	0.706	0.743	
Vehicle balanced	3	**0.727**	0.703	0.736	0.726	
	5	0.703	**0.71**	0.728	0.748	
	7	0.722	**0.723**	0.757	0.759	
	9	0.73	**0.743**	0.719	0.745	
	11	0.688	**0.714**	0.723	0.768	
Dry Bean imbalanced	3	0.909	**0.912**	0.906	0.92	
	5	0.908	**0.912**	0.902	0.918	
	7	0.908	**0.914**	0.903	0.917	
	9	0.906	**0.914**	0.904	0.915	
	11	0.906	**0.913**	0.903	0.914	
Dry Bean balanced	3	0.913	0.912	0.912	0.919	
	5	0.909	**0.916**	0.901	0.917	
	7	0.907	**0.912**	0.903	0.919	
	9	0.909	**0.913**	0.903	0.917	
	11	0.904	**0.912**	0.901	0.917	
Sensorless	3	**0.921**	0.92	0.95	0.949	
	5	**0.916**	0.911	0.945	0.934	
	7	**0.919**	0.915	0.947	0.942	
	9	**0.918**	0.917	0.943	0.942	
	11	**0.918**	0.914	0.942	0.941	
Crowd Sourced imbalanced	3	**0.827**	0.796	**0.838**	**0.838**	
	5	**0.84**	0.817	**0.875**	0.845	
	7	**0.851**	0.829	**0.855**	0.842	
	9	**0.851**	0.832	**0.859**	0.839	
	11	**0.846**	0.827	**0.876**	0.861	
Crowd Sourced balanced	3	**0.892**	0.864	**0.915**	0.879	
	5	**0.914**	0.874	**0.927**	0.894	
	7	**0.922**	0.864	**0.918**	0.899	
	9	**0.916**	0.86	**0.915**	0.893	
	11	**0.904**	0.861	**0.921**	0.903	

(*Continued*)

Table 23. (Continued)

Data set	No. tables	AVG 1HL	SUM 1HL	AVG 2HL	SUM 2HL	
Vehicle imbalanced	3	**0.718**	0.665	0.732	0.713	3AO
	5	**0.693**	0.671	0.711	0.72	
	7	0.686	**0.696**	0.702	0.724	
	9	0.694	**0.717**	0.686	0.723	
	11	**0.69**	0.688	0.698	0.739	
Vehicle balanced	3	**0.722**	0.677	0.76	0.743	
	5	**0.72**	0.717	0.732	0.734	
	7	0.732	**0.735**	0.74	0.765	
	9	0.702	**0.711**	0.728	0.741	
	11	0.692	**0.705**	0.71	0.747	
Dry Bean imbalanced	3	0.909	**0.911**	0.91	0.919	
	5	0.91	**0.913**	0.901	0.917	
	7	0.907	**0.913**	0.902	0.915	
	9	0.906	**0.913**	0.903	0.915	
	11	0.907	0.911	0.903	**0.908**	
Dry Bean balanced	3	**0.911**	**0.911**	0.918	0.92	
	5	0.91	**0.913**	0.901	0.919	
	7	0.907	**0.913**	0.902	0.918	
	9	0.906	**0.912**	0.904	0.916	
	11	0.905	**0.912**	0.904	0.92	
Sensorless	3	**0.92**	0.918	0.951	0.952	
	5	**0.914**	0.904	0.944	0.932	
	7	**0.919**	0.908	0.943	0.934	
	9	**0.916**	0.908	0.94	0.937	
	11	**0.922**	0.91	0.939	0.935	
Crowd Sourced imbalanced	3	**0.806**	**0.806**	**0.833**	0.806	
	5	**0.827**	0.802	**0.869**	0.838	
	7	**0.837**	0.818	**0.867**	0.834	
	9	**0.846**	0.808	**0.859**	0.844	
	11	**0.871**	0.842	**0.861**	0.838	
Crowd Sourced balanced	3	**0.833**	0.782	**0.858**	0.814	
	5	**0.892**	0.839	**0.919**	0.875	
	7	**0.916**	0.866	**0.918**	0.894	
	9	**0.921**	0.855	**0.916**	0.902	
	11	**0.914**	0.863	**0.923**	0.904	

https://doi.org/10.1371/journal.pone.0311041.t023

5.3 Comparison of classification quality for different numbers of hidden layers in the local and global networks

Table 23 also indicates a comparison of the results obtained for one and two hidden layers for each data set and settings analyzed. The better results are underlined. As can be seen, in twenty-seven cases the better results are obtained when using one hidden layer, while in one hundred and seventeen cases the better results are obtained when using two hidden layers. Thus, the use of two hidden layers generates better results in most cases. For statistical tests the

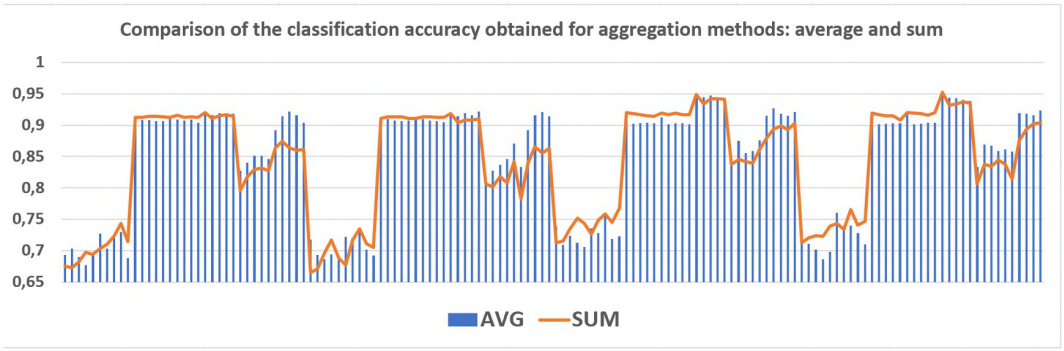


Fig 6. Comparison of classification accuracy *acc* obtained for different aggregation method.

https://doi.org/10.1371/journal.pone.0311041.g006

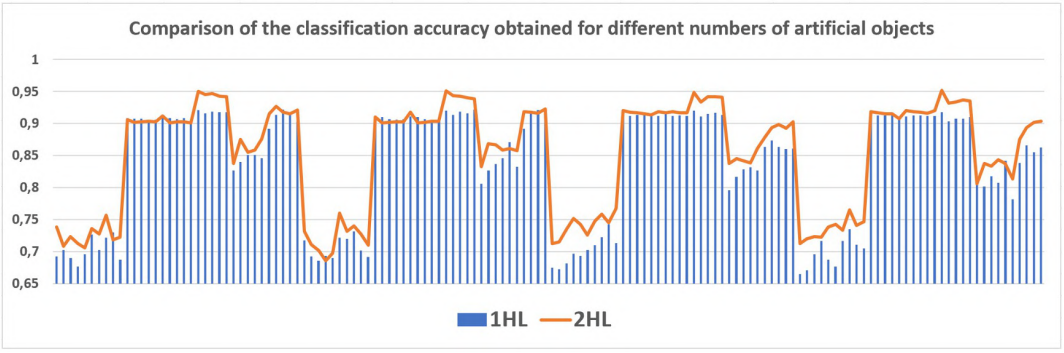


Fig 7. Comparison of classification accuracy *acc* obtained for different numbers of hidden layers.

https://doi.org/10.1371/journal.pone.0311041.g007

null hypothesis, H_θ is defined, whereby H_θ means there is no significance in terms of the method for aggregating local neural networks in the model. The Wilcoxon test for dependent samples confirmed the statistical significance of the differences in accuracy with $p = 0.0001$. Also, Fig 7 is created on which the results for all the settings and data sets analyzed are compared for one and two hidden layers. In the graph, it is evident the two hidden layers networks in many cases generates better results.

5.4 Comparison of classification quality of the proposed method versus other approaches

Table 24 shows all the results obtained for the proposed approach, different setting and all analyzed each data set. Based on the previous analyzes, it is concluded that in most cases, the best results are obtained with using one artificial object, the sum as aggregation method, and two hidden layers. But now, based on the summarized results in Table 24, the best approach and result is selected for each data set (marked in bold).

Table 25 shows the best obtained accuracy for the proposed approach and two known approaches from the literature: homogeneous ensemble of MLP network classifiers and ensemble of classifiers (KNN, DT, NB) with soft voting, which are described in detail in the previous section. The best result is shown in bold. The proposed method is the best which

Table 24. Comparison of classification accuracy *acc* obtained for the proposed method and approaches: 1AO-1HL-AVG, 3AO-1HL-AVG, 1AO-1HL-SUM, 3AO-1HL-SUM, 1AO-2HL-AVG, 3AO-2HL-AVG, 1AO-2HL-SUM, 3AO-2HL-SUM.

Data set	No. tables	Number of artificial objects generated based on one original object							
		1AO-	3AO-	1AO-	3AO-	1AO-	3AO-	1AO-	3AO-
		Number of hidden layers							
		1HL-	1HL-	1HL-	1HL-	2HL-	2HL-	2HL-	2HL-
		Aggregation method							
		AVG	AVG	SUM	SUM	AVG	AVG	SUM	SUM
Vehicle imbalanced	3	0.693	0.718	0.675	0.665	**0.739**	0.732	0.713	0.713
	5	0.703	0.693	0.673	0.671	0.709	0.711	0.715	**0.72**
	7	0.69	0.686	0.682	0.696	0.724	0.702	**0.735**	0.724
	9	0.677	0.694	0.697	0.717	0.713	0.686	**0.752**	0.723
	11	0.696	0.69	0.694	0.688	0.706	0.698	**0.743**	0.739
Vehicle balanced	3	0.727	0.722	0.703	0.677	0.736	**0.76**	0.726	0.743
	5	0.703	0.72	0.71	0.717	0.728	0.732	**0.748**	0.734
	7	0.722	0.732	0.723	0.735	0.757	0.74	0.759	**0.765**
	9	0.73	0.702	0.743	0.711	0.719	0.728	**0.745**	0.741
	11	0.688	0.692	0.714	0.705	0.723	0.71	**0.768**	0.747
Dry Bean imbalanced	3	0.909	0.909	0.912	0.911	0.906	0.91	**0.92**	0.919
	5	0.908	0.91	0.912	0.913	0.902	0.901	**0.918**	0.917
	7	0.908	0.907	0.914	0.913	0.903	0.902	**0.917**	0.915
	9	0.906	0.906	0.914	0.913	0.904	0.903	**0.915**	**0.915**
	11	0.906	0.907	0.913	0.911	0.903	0.903	**0.914**	0.908
Dry Bean balanced	3	0.913	0.911	0.912	0.911	0.912	0.918	0.919	**0.92**
	5	0.909	0.91	0.916	0.913	0.901	0.901	0.917	**0.919**
	7	0.907	0.907	0.912	0.913	0.903	0.902	**0.919**	0.918
	9	0.909	0.906	0.913	0.912	0.903	0.904	**0.917**	0.916
	11	0.904	0.905	0.912	0.912	0.901	0.904	0.917	**0.92**
Sensorless	3	0.921	0.92	0.92	0.918	0.95	0.951	0.949	**0.952**
	5	0.916	0.914	0.911	0.904	**0.945**	0.944	0.934	0.932
	7	0.919	0.919	0.915	0.908	**0.947**	0.943	0.942	0.934
	9	0.918	0.916	0.917	0.908	**0.943**	0.94	0.942	0.937
	11	0.918	0.922	0.914	0.91	**0.942**	0.939	0.941	0.935
Crowd Sourced imbalanced	3	0.827	0.806	0.796	0.806	**0.838**	0.833	**0.838**	0.806
	5	0.84	0.827	0.817	0.802	**0.875**	0.869	0.845	0.838
	7	0.851	0.837	0.829	0.818	0.855	**0.867**	0.842	0.834
	9	0.851	0.846	0.832	0.808	0.859	**0.859**	0.839	0.844
	11	0.846	0.871	0.827	0.842	**0.876**	0.861	0.861	0.838
Crowd Sourced balanced	3	0.892	0.833	0.864	0.782	**0.915**	0.858	0.879	0.814
	5	0.914	0.892	0.874	0.839	**0.927**	0.919	0.894	0.875
	7	**0.922**	0.916	0.864	0.866	0.918	0.918	0.899	0.894
	9	0.916	**0.921**	0.86	0.855	0.915	0.916	0.893	0.902
	11	0.904	0.914	0.861	0.863	0.921	**0.923**	0.903	0.904

https://doi.org/10.1371/journal.pone.0311041.t024

virtually always generates better results. Statistical tests are performed in order to confirm the importance in the differences in the obtained results *acc*. At first, the values of the classification accuracy in three dependent groups (proposed method, homogeneous ensemble of MPL and ensemble of classifiers KNN, DT, NB) are analyzed. For accuracy the Friedman statistics is

Table 25. Comparison of classification accuracy *acc* obtained for the proposed method and other methods known from the literature.

Dataset	No. tables	Proposed method	Homogeneous ensemble of MPL networks classifiers	Ensemble of classifiers (KNN, DT, NB)
Vehicle imbalanced	3	0.739	**0.74**	0.709
	5	0.72	**0.724**	0.709
	7	0.735	**0.74**	0.693
	9	**0.752**	0.685	0.717
	11	**0.743**	0.685	0.685
Vehicle balanced	3	**0.76**	0.756	0.732
	5	**0.748**	0.717	0.728
	7	**0.765**	0.728	0.752
	9	**0.745**	0.689	0.728
	11	**0.768**	0.685	0.677
Dry Bean imbalanced	3	**0.92**	0.91	0.906
	5	**0.918**	0.903	0.902
	7	**0.917**	0.901	0.899
	9	**0.915**	0.896	0.894
	11	**0.914**	0.901	0.9
Dry Bean balanced	3	**0.92**	0.911	0.909
	5	**0.919**	0.907	0.899
	7	**0.919**	0.905	0.9
	9	**0.917**	0.902	0.898
	11	**0.92**	0.904	0.903
Sensorless	3	**0.952**	0.93	0.888
	5	**0.945**	0.9	0.912
	7	**0.947**	0.878	0.897
	9	**0.943**	0.877	0.912
	11	**0.942**	0.86	0.926
Crowd Sourced imbalanced	3	0.838	0.86	**0.884**
	5	**0.875**	0.825	0.833
	7	**0.867**	0.797	0.809
	9	**0.859**	0.753	0.773
	11	**0.876**	0.735	0.763
Crowd Sourced balanced	3	**0.915**	0.879	0.866
	5	**0.927**	0.748	0.831
	7	**0.922**	0.74	0.782
	9	**0.921**	0.673	0.718
	11	**0.923**	0.634	0.682

https://doi.org/10.1371/journal.pone.0311041.t025

38.98 with df = 2, $p = 0.000001$ and we can again reject the null hypothesis. The average ranks are the following: Proposed approach 2.86; Homogeneous ensemble MLP 1.61; Ensemble of classifiers (KNN, DT, NB) 1.53. The critical value of difference of the Nemenyi test between the average ranks of two methods is 0.96. We can claim that classification accuracy of proposed approach is significantly better to all other classifiers (Fig 8). The Wilcoxon-each-pair test confirmed the significant differences between the average accuracy values for all pairs with p–value lower than 0.00002 between proposed method and the other analyzed. Also the post-hoc Dunn Bonferroni test confirmed that with $p = 0.000001$.

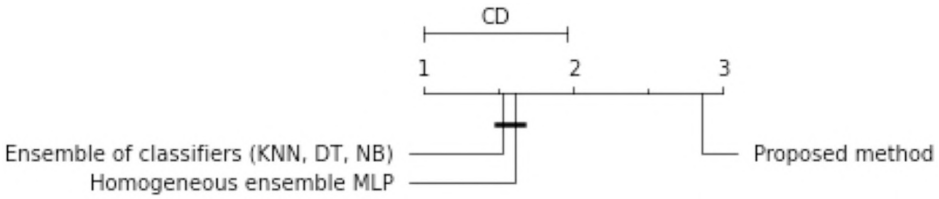


Fig 8. Critical value of difference (CD) and ranks for the Nemenyi test and accuracy values and methods: Proposed approach; Homogeneous ensemble MLP; Ensemble of classifiers (KNN, DT, NB). Groups of methods that are not significantly different (with the level of significance at 0.05) are connected.

https://doi.org/10.1371/journal.pone.0311041.g008

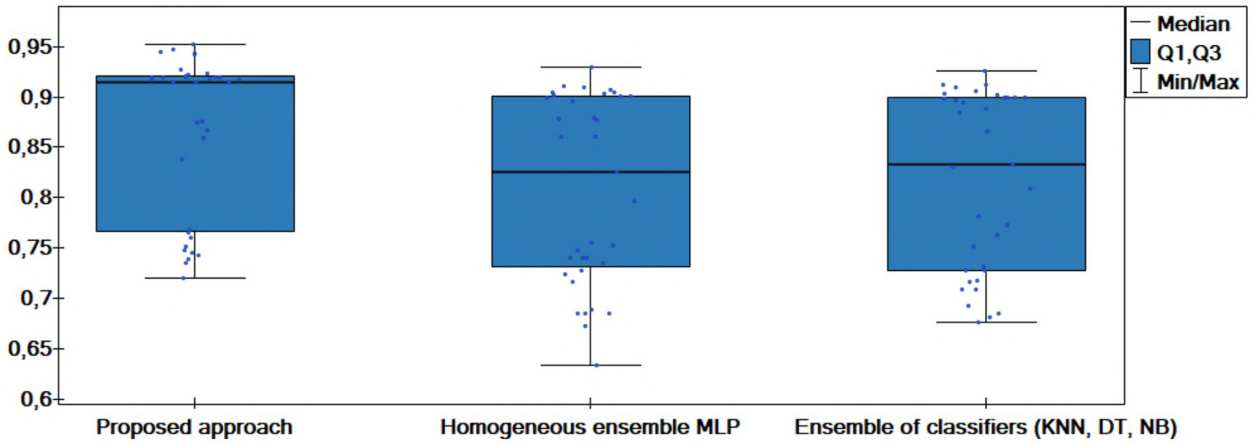


Fig 9. Box-plot chart with (Median, the first quartile—Q1, the third quartile—Q3) the value of classification accuracy *acc* for the proposed method and the other approaches.

https://doi.org/10.1371/journal.pone.0311041.g009

Additionally, comparative box-plot charts for the values of the classification accuracy and different approaches are created (Fig 9). As can be seen, the proposed approach generates by far the best quality of classification as this is confirmed by the highest positioned box plot and median.

In addition to classification accuracy, other measures are used for comparisons, which are good for unbalanced data and give more reliable comparisons. Table 26 shows the balanced accuracy for the proposed approach and homogeneous ensemble of MLP network classifiers and ensemble of classifiers (KNN, DT, NB) with soft voting. Balanced accuracy is calculated as the average of the sensitivity (true positive rate) for each class in a multiclass classification problem. The best result is shown in bold. Also for balanced accuracy, the proposed method yields the best results. Statistical tests are performed in order to confirm the importance in the differences in the obtained results *bacc*. At first, the values of the balanced accuracy in three dependent groups (proposed method, homogeneous ensemble of MPL and ensemble of classifiers KNN, DT, NB) are analyzed. For accuracy the Friedman statistics is 38 with df = 2, $p = 0.000001$ and we can again reject the null hypothesis. The average ranks are the following: Proposed approach 2.8; Homogeneous ensemble MLP 1.84; Ensemble of classifiers (KNN, DT, NB) 1.36. The critical value of difference of the Nemenyi test between the average ranks of two methods is 0.96. We can claim that classification accuracy of proposed approach is significantly better to all other classifiers (Fig 10). The Wilcoxon-each-pair test confirmed the

Table 26. Comparison of classification balanced accuracy *bacc* obtained for the proposed method and other methods known from the literature.

Data set	No. tables	Proposed method	Homogeneous ensemble of MPL networks classifiers	Ensemble of classifiers (KNN, DT, NB)
Vehicle imbalanced	3	0.709	**0.715**	0.686
	5	0.694	**0.722**	0.69
	7	0.708	**0.709**	0.67
	9	**0.719**	0.658	0.694
	11	**0.717**	0.66	0.672
Vehicle balanced	3	**0.731**	0.729	0.705
	5	**0.725**	0.661	0.698
	7	**0.732**	0.699	0.721
	9	**0.721**	0.649	0.7
	11	**0.746**	0.656	0.646
Dry Bean imbalanced	3	**0.931**	0.92	0.916
	5	**0.928**	0.908	0.908
	7	**0.928**	0.906	0.905
	9	**0.925**	0.9	0.897
	11	**0.924**	0.907	0.905
Dry Bean balanced	3	**0.931**	0.926	0.919
	5	**0.929**	0.919	0.907
	7	**0.93**	0.917	0.909
	9	**0.926**	0.913	0.907
	11	**0.929**	0.916	0.912
Sensorless	3	**0.952**	0.93	0.888
	5	**0.945**	0.9	0.912
	7	**0.947**	0.878	0.897
	9	**0.943**	0.877	0.912
	11	**0.942**	0.86	0.926
Crowd Sourced imbalanced	3	0.617	**0.856**	0.591
	5	0.847	0.804	0.49
	7	0.654	**0.801**	0.452
	9	0.645	**0.905**	0.389
	11	0.668	**0.728**	0.366
Crowd Sourced balanced	3	**0.915**	0.881	0.866
	5	**0.927**	0.793	0.831
	7	**0.922**	0.759	0.782
	9	**0.921**	0.71	0.718
	11	**0.923**	0.752	0.682

https://doi.org/10.1371/journal.pone.0311041.t026

significant differences between the average of balanced accuracy values for two approaches from the literature and the proposed approach with p–value lower than 0.004. The difference in average of balanced accuracy is not significant between the approaches the homogeneous ensemble of MLP network and the ensemble of classifiers (KNN, DT, NB). These conclusions are also confirmed graphically in Fig 11. Also the post-hoc Dunn Bonferroni test confirmed that with $p = 0.0002$.

The $F1$–score measure values are compared next (Table 27). This measure provides a balance between precision and recall. It helps evaluate the trade-off between making accurate

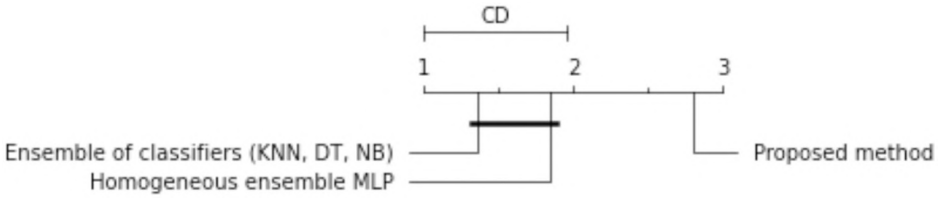


Fig 10. Critical value of difference (CD) and ranks for the Nemenyi test and balanced accuracy values and methods: Proposed approach; Homogeneous ensemble MLP; Ensemble of classifiers (KNN, DT, NB). Groups of methods that are not significantly different (with the level of significance at 0.05) are connected.

https://doi.org/10.1371/journal.pone.0311041.g010

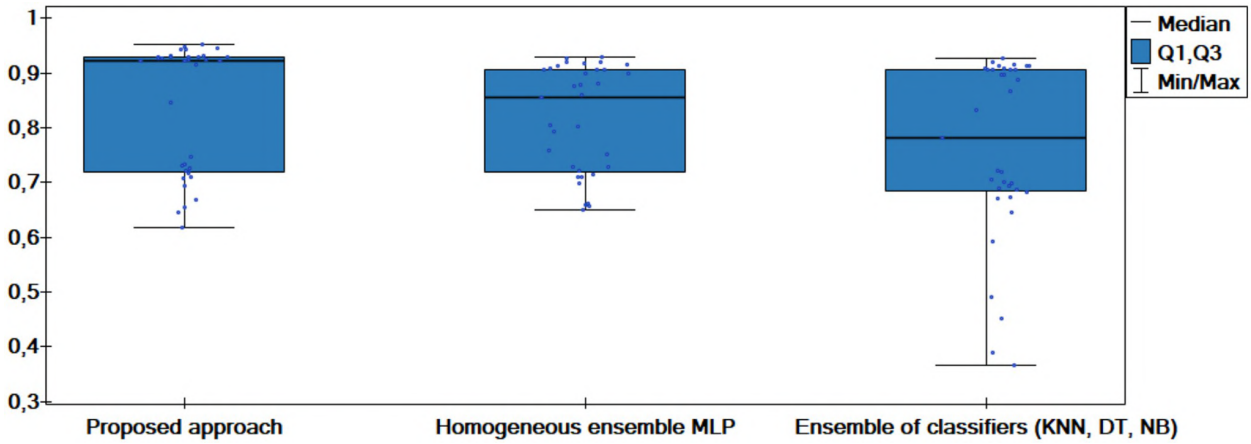


Fig 11. Box-plot chart with (Median, the first quartile—Q1, the third quartile—Q3) the value of balanced accuracy *bacc* for the proposed method and the other approaches.

https://doi.org/10.1371/journal.pone.0311041.g011

positive predictions (precision) and capturing all positive instances (recall). The *F*1−score is a good choice when you want to find a model that performs well in terms of both precision and recall. As we can see this time, the advantage of the proposed model over those from the literature is even greater. The Friedman test confirmed a statistically significant difference in the results obtained for the considered approaches, $\chi^2(35, 2) = 44.4$, $p = 0.000001$. The average ranks are the following: Proposed approach 2.91; Homogeneous ensemble MLP 1.63; Ensemble of classifiers (KNN, DT, NB) 1.46. The critical value of difference of the Nemenyi test between the average ranks of two methods is 0.96. We can claim that classification accuracy of proposed approach is significantly better to all other classifiers (Fig 12). The Wilcoxon-each-pair test confirmed the presence of significant differences in the average *F*1-score values between the two approaches from the literature and the proposed approach, with a *p*-value of less than 0.00001. These findings are visually reinforced by the data presented in Fig 13. Also the post-hoc Dunn Bonferroni test confirmed that with $p = 0.000001$.

In the last step, the precisions are compared. It quantifies the ability of a model to correctly identify positive instances while minimizing false positives. Precision is particularly important in scenarios where the cost of false positives is high, or when you want to ensure that the positive predictions made by the model are highly reliable. In Table 28 results are compared with the best score highlighted. Here, again, the proposed approach performs much better than the

Table 27. Comparison of classification *F*1−score obtained for the proposed method and other methods known from the literature.

Data set	No. tables	Proposed method	Homogeneous ensemble of MPL networks classifiers	Ensemble of classifiers (KNN, DT, NB)
Vehicle imbalanced	3	0.725	**0.728**	0.694
	5	**0.714**	0.695	0.696
	7	**0.727**	0.722	0.677
	9	**0.739**	0.652	0.699
	11	**0.732**	0.676	0.677
Vehicle balanced	3	**0.751**	0.746	0.716
	5	**0.73**	0.642	0.711
	7	**0.753**	0.708	0.733
	9	**0.743**	0.654	0.712
	11	**0.762**	0.662	0.65
Dry Bean imbalanced	3	**0.92**	0.91	0.906
	5	**0.918**	0.902	0.901
	7	**0.917**	0.9	0.899
	9	**0.915**	0.895	0.893
	11	**0.914**	0.901	0.9
Dry Bean balanced	3	**0.919**	0.911	0.909
	5	**0.919**	0.907	0.898
	7	**0.919**	0.905	0.899
	9	**0.916**	0.902	0.898
	11	**0.919**	0.904	0.902
Sensorless	3	**0.952**	0.93	0.881
	5	**0.944**	0.899	0.909
	7	**0.947**	0.877	0.894
	9	**0.943**	0.876	0.911
	11	**0.942**	0.859	0.926
Crowd Sourced imbalanced	3	0.84	0.867	**0.874**
	5	**0.872**	0.843	0.81
	7	**0.867**	0.827	0.78
	9	**0.859**	0.832	0.732
	11	**0.876**	0.824	0.713
Crowd Sourced balanced	3	**0.914**	0.879	0.866
	5	**0.927**	0.743	0.831
	7	**0.922**	0.742	0.78
	9	**0.921**	0.667	0.715
	11	**0.923**	0.62	0.678

https://doi.org/10.1371/journal.pone.0311041.t027

others. The Friedman test confirmed a statistically significant difference in the results obtained for the considered approaches, $\chi^2(35, 2) = 32.8$, $p = 0.000001$. Just as before, the Wilcoxon-each-pair test confirmed the presence of significant differences in the average precision values between the two approaches from the literature and the proposed approach, with a p-value of less than 0.0001. The post-hoc Dunn Bonferroni test confirmed that differences are significant between the proposed approach and the besline methods with $p = 0.000001$. The average ranks are the following: Proposed approach 2.76; Homogeneous ensemble MLP 1.8; Ensemble of classifiers (KNN, DT, NB) 1.44. The critical value of difference of the Nemenyi test between

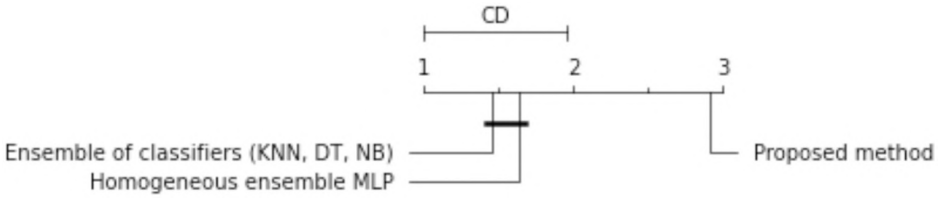


Fig 12. Critical value of difference (CD) and ranks for the Nemenyi test and F1-scorevalues and methods: Proposed approach; Homogeneous ensemble MLP; Ensemble of classifiers (KNN, DT, NB). Groups of methods that are not significantly different (with the level of significance at 0.05) are connected.

https://doi.org/10.1371/journal.pone.0311041.g012

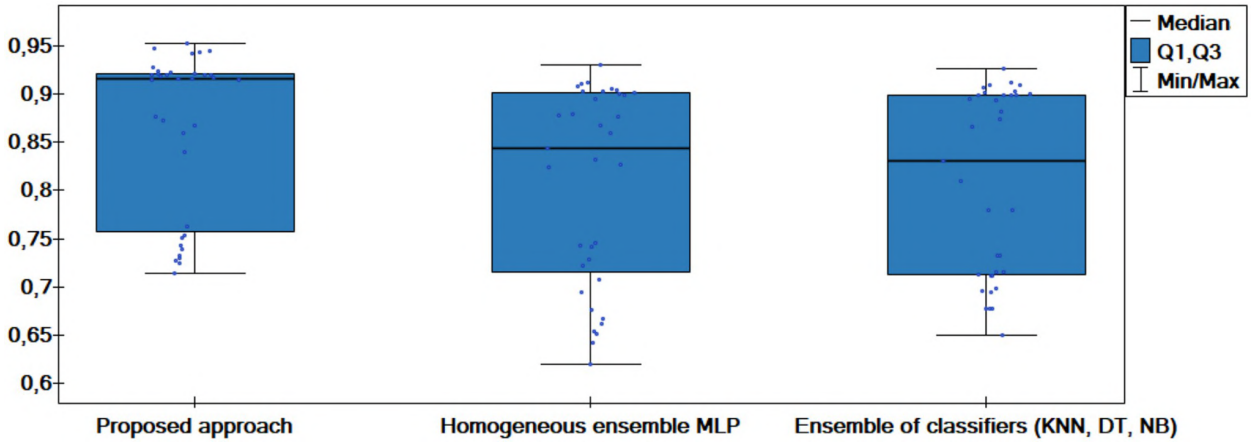


Fig 13. Box-plot chart with (Median, the first quartile—Q1, the third quartile—Q3) the value of *F*1−score for the proposed method and the other approaches.

https://doi.org/10.1371/journal.pone.0311041.g013

the average ranks of two methods is 0.96. We can claim that classification accuracy of proposed approach is significantly better to all other classifiers (Fig 14). The data presented in Fig 15 visually supports and reinforces these findings.

Based on the preceding analysis, it is clearly evident that the proposed approach consistently delivers very good results. To provide a more in-depth comparative analysis and demonstrate why the proposed model is superior, let us delve into the specific examples illustrating the advantages of the proposed approach. Let us notice once again that the proposed model involves training local MLP networks based on extended local tables (where missing values are imputed), aggregating these local models (using average or sum of weights), and re-training the global model with a shared sample of data. So the result is a single model that is more interpretable and easier to use. On the other hand, ensemble of classifiers (homogeneous MLP or heterogeneous KNN, DT, NB) involves creating base classifiers for each local table. The final decision is made by voting, so we do not get one model—one interpretation. The comparative analysis revealed that the proposed model consistently outperforms the baseline methods across all evaluation criteria. But the advantage of the proposed model obtained due to increased complexity is also justified practically. In the healthcare sector, predicting high-risk patients for sepsis across multiple hospitals is crucial for timely intervention and treatment. Each hospital has its own data set with various patient attributes, some of which may be missing or incomplete. With a single model obtained from the proposed approach, it is enough to

Table 28. Comparison of classification precision obtained for the proposed method and other methods known from the literature.

Data set	No. tables	Proposed method	Homogeneous ensemble of MPL networks classifiers	Ensemble of classifiers (KNN, DT, NB)
Vehicle imbalanced	3	**0.725**	**0.725**	0.701
	5	0.711	**0.761**	0.718
	7	**0.735**	0.73	0.69
	9	**0.746**	0.655	0.708
	11	**0.74**	0.67	0.705
Vehicle balanced	3	**0.756**	0.742	0.733
	5	**0.748**	0.582	0.726
	7	**0.755**	0.711	0.748
	9	**0.744**	0.66	0.718
	11	**0.768**	0.683	0.667
Dry Bean imbalanced	3	**0.921**	0.911	0.908
	5	**0.918**	0.903	0.904
	7	**0.917**	0.902	0.901
	9	**0.916**	0.898	0.897
	11	**0.915**	0.902	0.902
Dry Bean balanced	3	**0.92**	0.911	0.91
	5	**0.92**	0.907	0.9
	7	**0.919**	0.906	0.901
	9	**0.917**	0.903	0.899
	11	**0.92**	0.905	0.903
Sensorless	3	**0.953**	0.931	0.906
	5	**0.945**	0.9	0.921
	7	0.948	0.878	0.904
	9	**0.943**	0.879	0.918
	11	**0.943**	0.863	0.928
Crowd Sourced imbalanced	3	0.847	**0.886**	0.877
	5	0.874	**0.879**	0.818
	7	0.872	**0.889**	0.789
	9	0.868	**0.96**	0.74
	11	0.881	**0.969**	0.719
Crowd Sourced balanced	3	**0.915**	0.882	0.874
	5	**0.927**	0.772	0.837
	7	**0.923**	0.76	0.789
	9	**0.922**	0.684	0.726
	11	**0.924**	0.748	0.691

https://doi.org/10.1371/journal.pone.0311041.t028

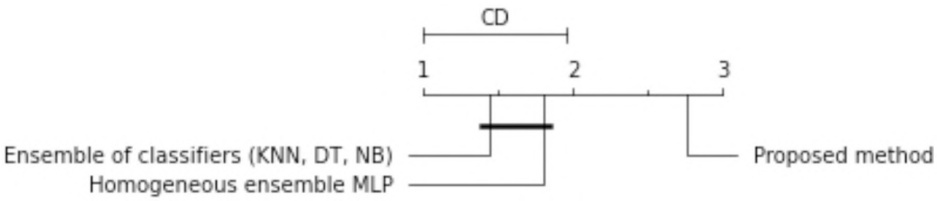


Fig 14. Critical value of difference (CD) and ranks for the Nemenyi test and precision values and methods: Proposed approach; Homogeneous ensemble MLP; Ensemble of classifiers (KNN, DT, NB). Groups of methods that are not significantly different (with the level of significance at 0.05) are connected.

https://doi.org/10.1371/journal.pone.0311041.g014

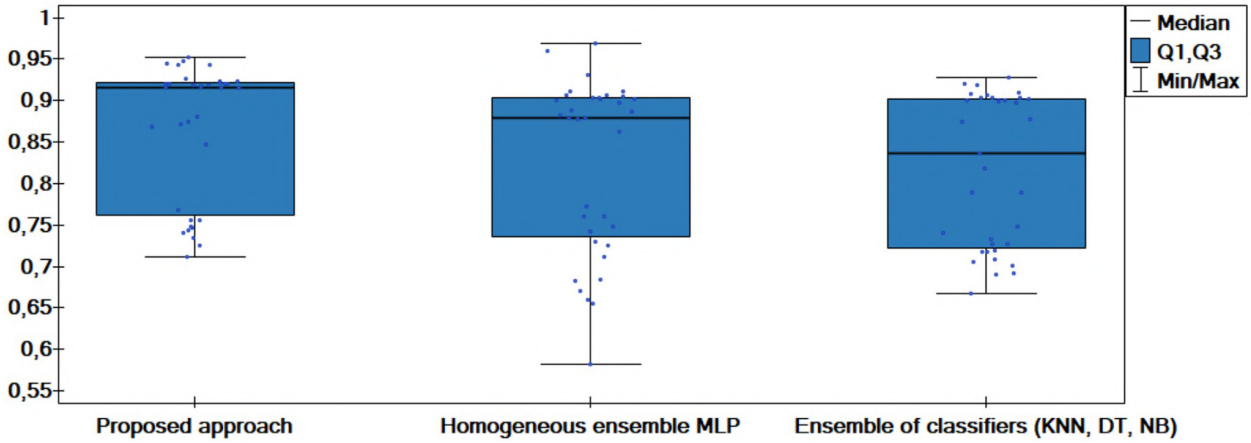


Fig 15. Box-plot chart with (Median, the first quartile—Q1, the third quartile—Q3) the value of precision for the proposed method and the other approaches.

https://doi.org/10.1371/journal.pone.0311041.g015

check all values on the attributes of the global model of the diagnosed patient—without having to refer to local hospitals and their databases. In smart agriculture, yield prediction is essential for effective farm management and planning. Multiple farms collect data on various attributes affecting crop yield, but these data sets can be incomplete or missing certain information. Also in this case the proposed model outperforms the baseline methods in integrating diverse and incomplete data from multiple farms to improve predictive accuracy, aiding farmers in making more informed decisions. These case studies illustrate the practical benefits and reliability of the proposed model in real-world applications, such as healthcare and agriculture, where accurate predictions are crucial for improving outcomes.

Additionally, AUCROC charts are prepared to demonstrate that the proposed approach outperforms those known from the literature. Due to limited space, we do not present the graphs for all thirty-five dispersed data sets. Figs 16 and 17 shows AUCROC graphs for Crowd Sourced imbalanced and balanced data sets, as well as all versions of dispersion—with 3, 5, 7, 9, and 11 local tables (since the proposed approach performed the worst for these data sets). Each row first displays the curve plot for the homogeneous ensemble of MPL networks classifiers, followed by the ensemble of classifiers (KNN, DT, NB), and finally, the proposed approach. Since the analyzed data sets are multi-class data, the graph shows the ROC curves for each decision class versus to the others, as well as the averaged ROC curve. It is clear that the proposed approach outperforms the other approaches. We can confidently say that for dispersed data, building a global neural network yields better results than using either heterogeneous or homogeneous ensembles of classifiers.

6 Conclusions

This paper proposes a new method for generating MLP global networks based on dispersed data with different sets of attributes. This method involves generating local MLP neural networks with identical structure based on local tables. Artificial objects based on the original objects are generated in order to realize that. In the next step, the networks are aggregated using appropriate weights proportional to the classification accuracy of the local models and one of two proposed methods—sum and average. The paper shows that the proposed model generated better results that other methods known from the literature. In addition, it is verified

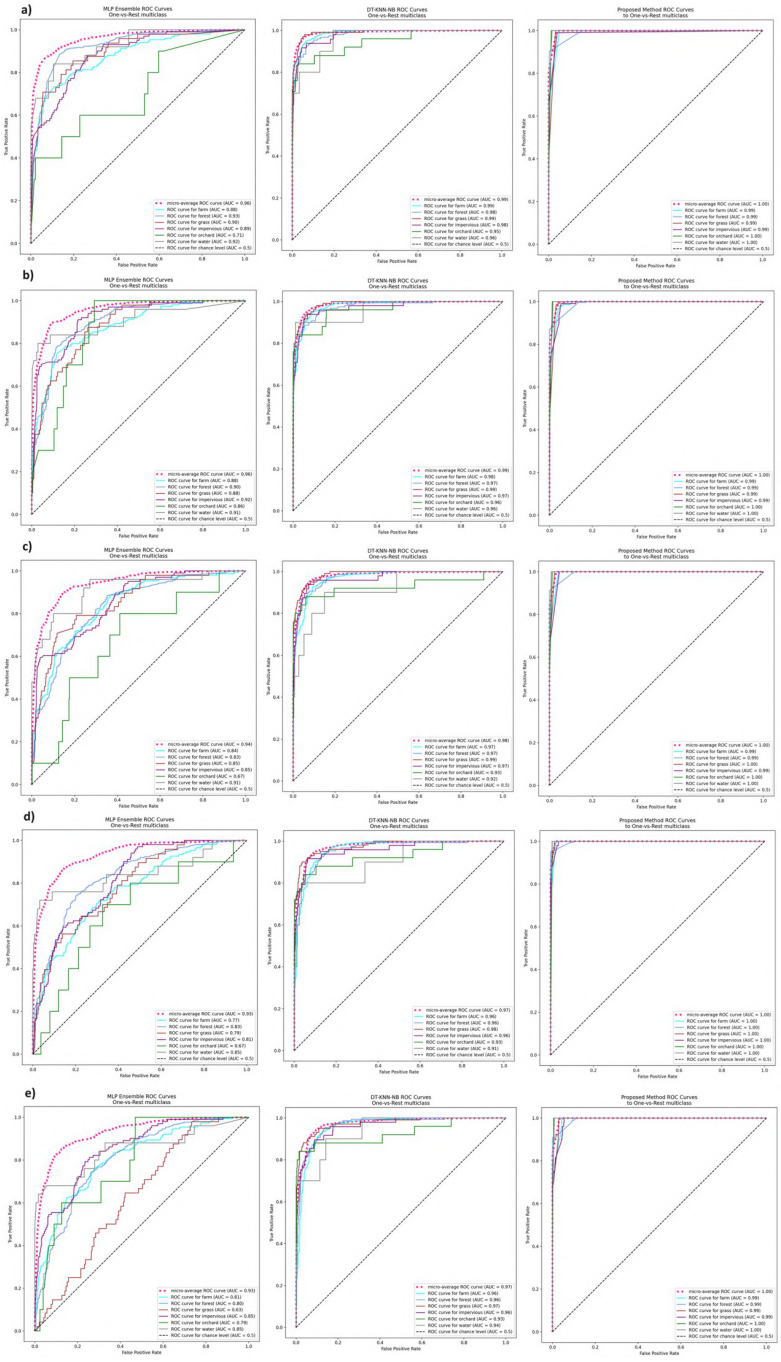


Fig 16. AUCROC graph for Crowd Sourced imbalanced data sets and all versions of dispersion: a) 3 local tables, b) 5 local tables, c) 7 local tables, d) 9 local tables, e) 11 local tables and three different approaches: first row graph—homogeneous ensemble of MPL networks classifiers, second row graph—ensemble of classifiers (KNN, DT, NB), third row graph—proposed approach.

https://doi.org/10.1371/journal.pone.0311041.g016

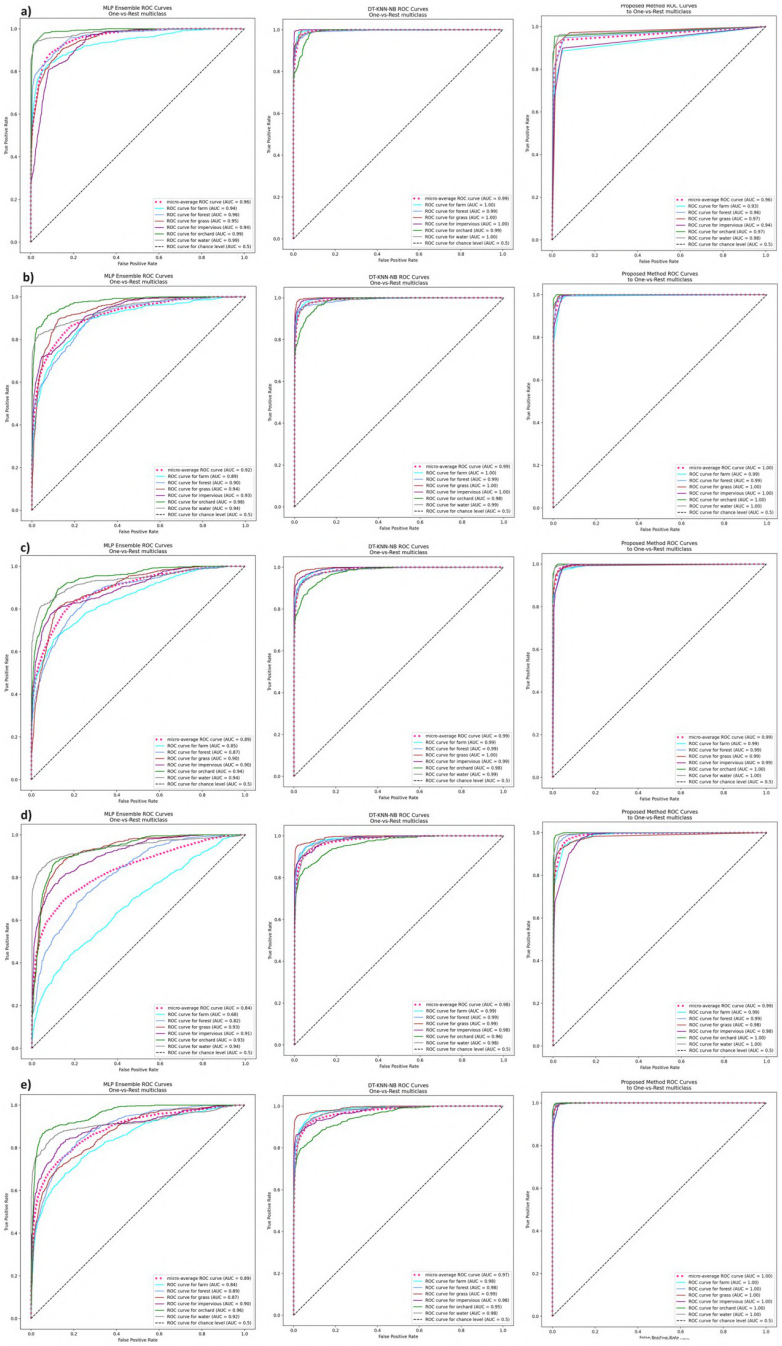


Fig 17. AUCROC graph for Crowd Sourced balanced data sets and all versions of dispersion: a) 3 local tables, b) 5 local tables, c) 7 local tables, d) 9 local tables, e) 11 local tables and three different approaches: first row graph—homogeneous ensemble of MPL networks classifiers, second row graph—ensemble of classifiers (KNN, DT, NB), third row graph—proposed approach.

https://doi.org/10.1371/journal.pone.0311041.g017

that on average, the best quality is achieved using only one artificial object with two hidden layers.

While the proposed model demonstrates significant improvements over traditional ensemble classifiers and homogeneous ensembles of MLPs, it is important to acknowledge certain limitations that could impact its performance and applicability in various scenarios. The model's ability to handle missing attributes heavily relies on the quality of the imputation methods used. If the imputation process introduces biases or inaccuracies, it can adversely affect the overall performance of the model. As the number of local tables increases, the model's ability to efficiently aggregate and re-train the global model might become a bottleneck. The model's performance is sensitive to the choice of parameters, such as the number of hidden layers and neurons, the method of aggregating local networks, and the strategies for handling missing data. Identifying the optimal settings requires extensive experimentation, which may not always be feasible. Developing automated parameter tuning methods or adaptive algorithms could mitigate this limitation. An important limitation of the method is also the need for a validation set, which must contain the combined characteristics of object—descriptions from the perspective of all local tables.

In further works, it is planned to use conflict analysis and coalitions of local networks to generate a global model as well as to develop a method for generating the artificial objects necessary for the global network's re-training stage. Also, it is planned to use other neural network architectures in the proposed approach.

Author Contributions

Conceptualization: Małgorzata Przybyła-Kasperek, Kwabena Frimpong Marfo.

Formal analysis: Małgorzata Przybyła-Kasperek, Kwabena Frimpong Marfo.

Investigation: Małgorzata Przybyła-Kasperek, Kwabena Frimpong Marfo.

Methodology: Małgorzata Przybyła-Kasperek, Kwabena Frimpong Marfo.

Project administration: Małgorzata Przybyła-Kasperek.

Software: Kwabena Frimpong Marfo.

Supervision: Małgorzata Przybyła-Kasperek.

Validation: Małgorzata Przybyła-Kasperek, Kwabena Frimpong Marfo.

Visualization: Małgorzata Przybyła-Kasperek, Kwabena Frimpong Marfo.

Writing – original draft: Małgorzata Przybyła-Kasperek.

Writing – review & editing: Małgorzata Przybyła-Kasperek, Kwabena Frimpong Marfo.

References

1. Li T., Sahu A., Talwalkar A., Smith V. Federated learning: Challenges, methods, and future directions. IEEE signal processing magazine. 2020; 37(3):50–60. https://doi.org/10.1109/MSP.2020.2973615
2. Verbraeken J., Wolting M., Katzy J., Kloppenburg J., Verbelen T., Rellermeyer J. A survey on distributed machine learning. Acm computing surveys (csur). 2020; 53(2):1–33. https://doi.org/10.1145/3377454
3. Kurian R., Lakshmi K. An ensemble classifier for the prediction of heart disease. International Journal of Scientific Research in Computer Science. 2018; 3(6):25–31.
4. Bilal A., Sun G., Mazhar S. Finger-vein recognition using a novel enhancement method with convolutional neural network. Journal of the Chinese Institute of Engineers. 2021; 44(5):407–417. https://doi.org/10.1080/02533839.2021.1919561

5. Bilal A., Liu X., Shafiq M., Ahmed Z., Long H. NIMEQ-SACNet: A novel self-attention precision medicine model for vision-threatening diabetic retinopathy using image data. Computers in Biology and Medicine. 2024; 171:108099. https://doi.org/10.1016/j.compbiomed.2024.108099 PMID: 38364659
6. Feng X., Xiu Y., Long H., Wang Z., Bilal A., Yang L. Advancing single-cell RNA-seq data analysis through the fusion of multi-layer perceptron and graph neural network. Briefings in Bioinformatics. 2024; 25(1):. https://doi.org/10.1093/bib/bbad481
7. Mendoza J., Pedrini H. Detection and classification of lung nodules in chest X-ray images using deep convolutional neural networks. Computational Intelligence. 2020; 36(2):370–401. https://doi.org/10.1111/coin.12241
8. Bilal A., Sun G., Mazhar S., Junjie Z. Neuro-optimized numerical treatment of HIV infection model. International Journal of Biomathematics. 2021; 14(05):2150033. https://doi.org/10.1142/S1793524521500339
9. Bilal A., Liu X., Long H., Shafiq M., Waqar M. Increasing Crop Quality and Yield with a Machine Learning-Based Crop Monitoring System. Computers, Materials & Continua. 2023; 76(2):.
10. Yu L., Li M. A case-based reasoning driven ensemble learning paradigm for financial distress prediction with missing data. Applied Soft Computing. 2023; 137:110163. https://doi.org/10.1016/j.asoc.2023.110163
11. Kang J., Ullah Z., Gwak J. MRI-based brain tumor classification using ensemble of deep features and machine learning classifiers. Sensors. 2021; 21(6):2222. https://doi.org/10.3390/s21062222 PMID: 33810176
12. Sesmero M., Iglesias J., Magán E., Ledezma A., Sanchis A. Impact of the learners diversity and combination method on the generation of heterogeneous classifier ensembles. Applied Soft Computing. 2021; 111:107689. https://doi.org/10.1016/j.asoc.2021.107689
13. Arora J., Agrawal U., Tiwari P., Gupta D., Khanna A. Ensemble feature selection method based on recently developed nature-inspired algorithms. In: International Conference on Innovative Computing and Communications: Proceedings of ICICC 2019, Volume 1. Springer; 2020. p. 457–470. 2020.
14. Yaiprasert C., Hidayanto A. AI-driven ensemble three machine learning to enhance digital marketing strategies in the food delivery business. Intelligent Systems with Applications. 2023; 18:200235. https://doi.org/10.1016/j.iswa.2023.200235
15. Bhat P., Behal S., Dutta K. A system call-based android malware detection approach with homogeneous & heterogeneous ensemble machine learning. Computers & Security. 2023; 130:103277. https://doi.org/10.1016/j.cose.2023.103277
16. Dinkel H., Wang Y., Yan Z., Zhang J., Wang Y. CED: Consistent ensemble distillation for audio tagging. In: ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE; 2024. p. 291–295. 2024.
17. Mothukuri V., Parizi R., Pouriyeh S., Huang Y., Dehghantanha A., Srivastava G. A survey on security and privacy of federated learning. Future Generation Computer Systems. 2021; 115:619–640. https://doi.org/10.1016/j.future.2020.10.007
18. Bazan J., Milan P., Bazan-Socha S., Wójcik K. Application of Federated Learning to Prediction of Patient Mortality in Vasculitis Disease. In: International Joint Conference on Rough Sets. Springer; 2023. p. 526–536. 2023.
19. Li Z., Lin T., Shang X., Wu C. Revisiting weighted aggregation in federated learning with neural networks. In: International Conference on Machine Learning. PMLR; 2023. p. 19767–19788. 2023.
20. Zhu H., Zhang H., Jin Y. From federated learning to federated neural architecture search: a survey. Complex & Intelligent Systems. 2021; 7:639–657. https://doi.org/10.1007/s40747-020-00247-z
21. Alazab M., Priya R. M. P., Maddikunta P., Gadekallu T., Pham Q. Federated Learning for Cybersecurity: Concepts, Challenges, and Future Directions. IEEE Trans. Ind. Informatics. 2022; 18(5):3501–3509. https://doi.org/10.1109/TII.2021.3119038
22. Dyczkowski K., Pekala B., Szkoła J., Wilbik A. Federated learning with uncertainty on the example of a medical data. In: 2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). IEEE; 2022. p. 1–8. 2022.
23. Singhal K., Sidahmed H., Garrett Z., Wu S., Rush J., Prakash S. Federated reconstruction: Partially local federated learning. Advances in Neural Information Processing Systems. 2021; 34:11220–11232.
24. Guo R., Shen W. A model fusion method for online state of charge and state of power co-estimation of lithium-ion batteries in electric vehicles. IEEE Transactions on Vehicular Technology. 2022; 71(11):11515–11525. https://doi.org/10.1109/TVT.2022.3193735
25. Marfo K., Przybyła-Kasperek M. Radial basis function network for aggregating predictions of k-nearest neighbors local models generated based on independent data sets. Procedia Computer Science. 2022; 207:3234–3243. https://doi.org/10.1016/j.procs.2022.09.381

26. Moshkov M. Common Decision Trees, Rules, and Tests (Reducts) for Dispersed Decision Tables. Procedia Computer Science. 2022; 207:2503–2507. https://doi.org/10.1016/j.procs.2022.09.308
27. Przybyła-Kasperek M., Aning S. Bagging and single decision tree approaches to dispersed data. In: Computational Science–ICCS 2021: 21st International Conference, Krakow, Poland, June 16–18, 2021, Proceedings, Part III. Springer; 2021. p. 420–427. 2021.
28. Przybyła-Kasperek M., Marfo K. Neural network used for the fusion of predictions obtained by the K-nearest neighbors algorithm based on independent data sources. Entropy. 2021; 23(12):1568. https://doi.org/10.3390/e23121568 PMID: 34945874
29. Czarnowski I. Weighted Ensemble with one-class Classification and Over-sampling and Instance selection (WECOI): An approach for learning from imbalanced data streams. Journal of Computational Science. 2022; 61:101614. https://doi.org/10.1016/j.jocs.2022.101614
30. Przybyła-Kasperek M. The power of agents in a dispersed system–The Shapley-Shubik power index. Journal of Parallel and Distributed Computing. 2021; 157:105–124. https://doi.org/10.1016/j.jpdc.2021.06.010
31. Przybyła-Kasperek M., Kusztal K. New Classification Method for Independent Data Sources Using Pawlak Conflict Model and Decision Trees. Entropy. 2022; 24(11):1604. https://doi.org/10.3390/e24111604 PMID: 36359694
32. Elshamy R., Abu-Elnasr O., Elhoseny M., Elmougy S. Improving the efficiency of RMSProp optimizer by utilizing Nestrove in deep learning. Scientific Reports. 2023; 13(1):8814. https://doi.org/10.1038/s41598-023-35663-x PMID: 37258633
33. Stephen A., Punitha A., Chandrasekar A. Designing self attention-based ResNet architecture for rice leaf disease classification. Neural Computing and Applications. 2023; 35(9):6737–6751. https://doi.org/10.1007/s00521-022-07793-2
34. Qureshi A., Roos T. Transfer learning with ensembles of deep neural networks for skin cancer detection in imbalanced data sets. Neural Processing Letters. 2023; 55(4):4461–4479. https://doi.org/10.1007/s11063-022-11049-4
35. Bishop C. Pattern recognition and machine learning, 5th Edition. Information science and statistics. Springer 2007.
36. Glorot X., Bordes A., Bengio Y. Deep Sparse Rectifier Neural Networks. In: Gordon GJ, Dunson DB, Dudik M, editors. Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011. vol. 15 of JMLR Proceedings. JMLR.org; p. 315–323. 2011.
37. Li X., Li X., Pan D., Zhu D. On the learning property of logistic and softmax losses for deep neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34; p. 4739–4746. 2020.
38. Kingma D., Ba J. Adam: A Method for Stochastic Optimization. In: Bengio Y, LeCun Y, editors. 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings 2015.
39. Mannor S., Peleg D., Rubinstein R. The cross entropy method for classification. In: Raedt LD, Wrobel S, editors. Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005. vol. 119 of ACM International Conference Proceeding Series. ACM; 2005. p. 561–568. 2005.
40. Schapire R. Explaining adaboost. Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik. 2013;:37–52.
41. Hemachandra A., Dai Z., Singh J., Ng S., Low B. Training-free neural active learning with initialization-robustness guarantees. In: International Conference on Machine Learning. PMLR; 2023. p. 12931–12971 2023.
42. Saran A., Yousefi S., Krishnamurthy A., Langford J., Ash J. Streaming active learning with deep neural networks. In: International Conference on Machine Learning. PMLR; 2023. p. 30005–30021 2023.
43. Zamri N., Azhar S., Mansor M., Alway A., Kasihmuddin M. Weighted Random k Satisfiability for k = 1,2 (r2SAT) in Discrete Hopfield Neural Network. Applied Soft Computing. 2022; 126:109312. https://doi.org/10.1016/j.asoc.2022.109312
44. Zamri N., Azhar S., Sidik S., Mansor M., Kasihmuddin M., Pakruddin S., et al. Multi-discrete genetic algorithm in hopfield neural network with weighted random k satisfiability. Neural Computing and Applications. 2022; 34(21):19283–19311. https://doi.org/10.1007/s00521-022-07541-6
45. Siebert J. Vehicle recognition using rule based methods. Turing Institute Research Memorandum. 1987;TIRM-87-0.18:.
46. Koklu M., Özkan I. Multiclass classification of dry beans using computer vision and machine learning techniques. Comput. Electron. Agric. 2020; 174:105507. https://doi.org/10.1016/j.compag.2020.105507

47. Bator M., Wissel C., Dicks A., Lohweg V. Feature Extraction for a Conditioning Monitoring System in a Bottling Process. In: 23rd IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2018, Torino, Italy, September 4-7, 2018. IEEE; 2018. p. 1201–1204. 2018.

48. Johnson B. Crowdsourced Mapping. UCI Machine Learning Repository, 2016.

49. Chawla N., Bowyer K., Hall L., Kegelmeyer W. SMOTE: Synthetic Minority Over-sampling Technique. J. Artif. Intell. Res. 2002; 16:321–357. https://doi.org/10.1613/jair.953

50. Russell I., Markov Z. An introduction to the Weka data mining system. 2017.

51. Zamri N., Mansor M., Kasihmuddin M., Sidik S., Alway A., Romli N., Guo Y., et al. A modified reverse-based analysis logic mining model with Weighted Random 2 Satisfiability logic in Discrete Hopfield Neural Network and multi-objective training of Modified Niched Genetic Algorithm. Expert Systems with Applications. 2024; 240:122307. https://doi.org/10.1016/j.eswa.2023.122307

Publication [P10]

Marfo K.F.,Przybyła-Kasperek M. Objects Diversity and its Impact on Classification Quality in Dispersed Data Environments *Vietnam Journal of Computer Science*, SN: 2196-8888 p253, 2025.
URL: https://doi.org/10.1142/S2196888824500180.
DOI: 10.1142/s2196888824500180.
$\text{MEiN}_{2025} = 20$
Number of citations:

- according to Web of Science: 0
- according to Google Scholar: 0

Contributor	Description of main tasks
Kwabena Frimpong Marfo	- investigation - conceptualization and methodology - result analysis - implementation of proposed algorithm - manuscript: review and editing - visualization: creating all figures in manuscript
Małgorzata Przybyła-Kasperek	- conceptualization and methodology - result analysis - manuscript: original draft preparation - manuscript: review and editing - visualization: creating all figures in manuscript

OPEN ACCESS
Vietnam Journal of Computer Science
Vol. 12, No. 3 (2025) 253–275

DOI: 10.1142/S2196888824500180

Objects Diversity and its Impact on Classification Quality in Dispersed Data Environments

Kwabena Frimpong Marfo* and Małgorzata Przybyła-Kasperek†

Institute of Computer Science, University of Silesia in Katowice
Będzińska 39, 41-200 Sosnowiec, Poland
**kwabena.marfo@us.edu.pl*
†*malgorzata.przybyla-kasperek@us.edu.pl*

Accepted 12 October 2024
Published 19 November 2024

This paper studies a classification model dedicated to dispersed data, employing the k-nearest neighbors method as a base classifier and a Radial Basis Function (RBF) network as a fusion method. Focusing on classifying objects described by different attributes, the study systematically reduces common objects in local tables to assess the robustness of the model. Surprisingly, the proposed approach shows resilience in reducing common objects without significantly affecting key metrics such as F-measure, balanced accuracy and overall accuracy. Moreover, the studied model performs better on balanced data. This research contributes valuable insights into dispersed data classification, demonstrating the model's effectiveness in handling diverse objects and attributes. The findings have implications for fields reliant on dispersed data storage, such as healthcare, banking, and surveillance, showcasing the model's potential for real-world applications.

Keywords: Dispersed data; radial basis function network; reduced common objects.

1. Introduction

In today's world, the presence of data-driven decision-making problems appears nearly on every facet of our lives. With the growth in data collection capabilities, the way of gathering data has gradually shifted towards dispersed data storage architectures. This approach entails the decentralized storage of data across various sources — a strategy employed in many fields such as healthcare, banking and surveillance. While dispersed data storage offers unparalleled advantages, it introduces a huge challenge — the harmonization of dispersed data with different

† Corresponding author.

features describing the same objects. Objects of interest, be they medical records, images captured from different viewpoints, or any other form of data are often represented in a fragmented manner across data silos, or sometimes, only a fraction of the features describing a given object is stored in a local table.[27,28] This dispersion arises from the use of various sensors, data collection methods, or simply the decentralized nature of data source systems.

Fundamental to this challenge is the need to overcome the inconsistencies and differences in the way objects are represented locally. When there are different attributes and objects in local tables, achieving a uniform structure across different datasets is a non-trivial task. To address this challenge, this paper proposes an approach that emphasizes on treating local data in isolation. Rather than imposing a global structure, the focus is on building local models that distinguish patterns in each dataset independently. The synthesis of these local models, facilitated by neural networks, more specifically an RBF network, allows the identification of overarching patterns that transcend the local context. Neural networks, with their inherent ability to learn complex patterns and relationships, are emerging as a promising tool in this respect.

The goal of this study is to investigate the possibility of building some general patterns that will provide good classification quality based on different objects described by a variety of attributes in local tables. To achieve this, a two-stage dispersion is considered in terms of both objects and attributes. In this study, we test the classification quality of a fusion model which comprises the k-nearest neighbors and a RBF network on dispersed data with diversified objects. We examine the results with gradually increasing object diversity and with dispersed attributes included in individual local tables.

Thus, the paper's contribution is to propose the model specifically addresses the challenge of dispersed data storage using the k-nearest neighbors method as a base classifier and a RBF network as the fusion method. A unique contribution of this research is its focus on object diversity — where objects in local datasets are represented by different attributes (but some can be shared) — and how the model performs when the number of shared objects between datasets is reduced. The paper's findings suggest that the model is resilient to reducing common objects, maintaining classification quality through metrics such as F-measure, balanced accuracy, and overall accuracy.

The research procedure (research steps) is as follows. We use data with dispersed objects and attributes for the model, which utilizes k-nearest neighbors as the base classifier and an RBF network for fusion. The study uses five distinct datasets from open repositories, generating dispersed decision tables by randomly distributing attributes and objects across multiple local tables. The dispersion is examined under varying percentages of common objects (15%, 30%, 45%, 60%) and different levels of data imbalance (imbalanced and balanced using the SMOTE method). The experiments involve generating predictions from local tables and testing the model with various k values in k-nearest neighbors (1, 5, 10) and different numbers of

neurons in the hidden layer of the RBF network. The classification performance is evaluated using multiple metrics (Precision, Recall, F-measure, balanced accuracy, accuracy) on balanced and imbalanced datasets. The robustness of the model to object diversity is analyzed statistically. Tests like the Friedman test and the Wilcoxon test are used to evaluate the effect of object diversity, balancing of datasets, and parameter tuning on classification performance. The results show that the model performs well despite reduced commonality of objects across tables, and balancing the data improves classification quality. The experiments conclude that the proposed model is resistant to object diversity and performs better on balanced data.

This paper is an extended version of the paper "Exploring the Impact of Object Diversity on Classification Quality in Dispersed Data Environments" presented at 16th Asian Conference on Intelligent Information and Database Systems ACIIDS 2024.[1] In this paper, the following parts have been added compared to the conference paper:

- the literature review has been extended,
- the experiments have been expanded to include an additional diversification of objects percentage (60%), which allows to study more deeply the impact of objects diversity in local tables on classification quality,
- experiments for two additional datasets are presented,
- more sophisticated and expanded statistical analysis have been performed and additional conclusions have been drawn based on this.

This paper is organized as follows. The literature review is presented in Sec. 2. In Sec. 3, the proposed classification model using an RBF neural network is described and the method for generating dispersed data with a reduced number of common objects. Section 4 addresses the datasets that are used and presents the conducted experiments and discussion on obtained results. Section 5 is on conclusions and future research plans.

2. Literature Review

The research on classification models for dispersed data has become increasingly significant due to the growing complexity and decentralization of data across various domains. Dispersed data, characterized by its autonomous collection and storage in different repositories, present unique challenges that traditional machine learning methods often struggle to address effectively.

Effective approach to handling dispersed data is to employ ensemble learning techniques. The fundamental concept revolves around the construction of a collection of classifiers from which joint decisions are made by merging these classifiers. The most recent advancements in this field can be found in the paper.[2] The utilization of ensemble classifiers can be found in various domains with notable applications in data stream classification.[3,4] These methodologies also

enable the handling of large datasets from social networks or mobile devices.[5] Moreover, the application of ensemble learning in the realm of economics is explored in Ref. 6. Diverse techniques have been explored for recognizing objects from fragmented or partial data.[7] The creation of a classifier group can be achieved by training distinct learning algorithms on the same dataset or by generating multiple training sets. Techniques for ensemble-based classification can be categorized based on how training sets are generated such as through dataset partitioning like bagging or partitioning the attribute space as exemplified by the Random Forest approach.[8,9] The connection of classifier predictions is typically carried out using fusion methods, often leveraging weighting techniques or probabilistic strategies.[10,11] In addition to classifier combination, another approach involves classifier selection methods.[12]

In addition to classifier ensembles, other approaches to fragmented data can be found in the literature. Among them are hidden Markov models which were employed for gesture recognition based on fragmentary vision in Ref. 13, showcasing the adaptability of different methodologies to handle fragmented information. Also, fuzzy rules were utilized for fragmented handwritten digit recognition in Ref. 14, highlighting the versatility of rule-based systems in handling partial information. Addressing the challenges posed by data isolation and privacy concerns, Federated Learning (FL) has emerged as a privacy-preserving paradigm.[15] FL enables collaborative model training without the need to centralize sensitive data, making it particularly well-suited for scenarios involving dispersed data. The applicability of FL extends to fields such as healthcare, where patient records are dispersed across multiple hospitals, each maintaining its data silo. The objective is to achieve a unified prediction utilizing all accessible datasets. Typically, individual models are constructed based on specific datasets, which are then combined. In federated learning, ensuring data security and privacy is crucial.[16]

While this paper also employs a group of classifiers but it differs in its assumptions compared to the above approaches. The main difference lies in the dispersion of both objects and features. In addition, the paper examines whether object diversification affects the classification quality of the approach proposed in Ref. 17. Fusion methods, as discussed in Ref. 17, employ k-nearest neighbors and Multi-Layer Perceptron (MLP) classifiers for dispersed data classification. While achieving favorable results, these methods do not scale well with noisy data. Perturbations in noise intensity and the number of neurons in the hidden layer significantly impact performance. Also, the above study considered homogeneous objects in local tables and did not account for the diversification of objects that could occur in local tables. In the present study, this is precisely the approach being considered.

In this paper, we significantly extend the research contained in the conference paper[1] with additional comparisons with a different percentage of common objects and additional datasets investigated, statistical tests, and graphs.

3. Materials and Methods

In the context of dispersed data storage, the process of constructing a unified structure of local data is a big challenge due to inherent data inconsistencies. To overcome this, an approach that treats local data in isolation is adopted. Patterns within each set are discovered by building local models, after which possible additional patterns in the predictions of the local models are sought out for using a neural network.

Consider a scenario where we have access to local decision tables denoted by

$$D_i = (U_i, A_i, d) \tag{1}$$

for $i \in 1, \ldots, n$. Here, U_i constitutes the universe, comprising a set of objects; A_i encompasses conditional attributes — features describing the objects; and d represents the decision attribute — labels. It is worth noting that objects and attributes in local tables may vary, with potential common elements being present among them.

This dispersion reflects situation in which independent sensors capture different features (some of them can be shared) on various objects in order to capture common patterns. It should be noted that the goal of these sensors is common — to make a decision on the same labels. In practical terms, such data can be found in dispersed medical data from many hospitals where one would want to make a prediction on the organ affected by cancer, or data from multiple cameras that capture objects from different perspectives in order to extract some general characteristics of the object. It is obvious that attributes differ in such an approach due to dispersion. In previous study,[18,19] a classification method based on dispersed data using the k-nearest neighbors classifier as the base classifier and a RBF neural network as the fusion method was proposed. This approach produced good results, however what it lacked was the diversification of objects, as objects in local tables were the same — only the attributes were dispersed. In the previous papers,[18,19] in order to model as close to a real situation as possible, object identifiers in local tables were not stored and recognized to generate global decisions, so aggregation of objects was not possible in any way. However, the question remains — whether uniformity/homogeneity of objects in local tables is necessary to achieve high quality in such a dispersed model. Perhaps, if the objects could also be different in local tables, one could analyze different patients in hospitals (which is very realistic) or show different object fragments to cameras and still retrieve patterns at the appropriate level of accuracy. In this study, we examine, for the first time, the effect of object diversity occurring in local tables on the obtained classification results. The robustness of the system against the presence of completely different objects between local tables is tested by gradually decreasing the percentage of occurring common objects between tables.

As mentioned in the papers,[18,19] local models are built based on local tables. Various models for building local classifiers can be used, however, in the previous research as well as in this paper, the k-nearest neighbors classifier was chosen for its

low computational complexity and suitability for classification based on fragmented features. The expectation is for objects of the same type to exhibit similar features. To compute distances between test objects and object in local tables, the Gower measure is employed.[20] This measure can be used for attributes of different types (quantitative, qualitative, binary) and from different ranges in the decision table, eliminating the need for normalization or standardization. In the classification process, each base classifier performs classification using a subset of attributes that are present in the given local tables — a classifier i predicting on a decision table D_i utilizes the set of features A_i. It is crucial to note that the test objects must possess all features occurring in local tables. A classifier i produces a probability vector over decision classes for a test object x (denoted as $\mu_i(x)$). The vector's dimension is equal to the number of decision classes. Each coefficient $\mu_{i,j}(x)$ is determined using the k-nearest neighbors of the test object x belonging to a given decision class j and decision table D_i.

The challenging task of identifying the correct decision class for the test object is addressed using an RBF neural network. This approach has already yielded good results in previous papers[18,19] and has proven successful in aggregating predictions generated from dispersed data when the same objects were stored in each local table, whereas the main issue analyzed now is the robustness of this model against the variation of objects in local tables. RBF networks have shown better classification quality than MLP networks so we focus on this approach in this paper. However, previous studies did not consider a two-stage dispersion in terms of both objects and attributes, which is considered in this paper for the first time.

RBF network has the ability to transform input vectors into a new feature space where classes become linearly separable. For this transformation to be effective, the hidden layer must possess more neurons than the input layer. RBF neural networks inherently consist of only a hidden layer. In our scenario, the input vector is constructed from the prediction vectors generated by the base classifiers. Formally, a vector

$$[\mu_1(x), \ldots, \mu_n(x)] \tag{2}$$

is created for the test object x, with a dimension of $n \cdot c$ since there are n base classifiers, each generating a c-dimensional vector, where c is the number of decision classes. The input layer of the network consists of $n \cdot c$ neurons. Each neuron in the hidden layer is characterized by a parameter called the center. Formally, the kth neuron in the hidden layer possesses the center c_k, interpreted as a representative of a specific group of objects (here it will be a specific group of prediction combinations). RBF networks reduces the influence of distant input on the network's output while incorporating them into the classification process. The back-propagation algorithm is employed to train the networks. In the experimental part, a stratified 10-fold cross-validation method is employed to train the RBF network on the test set. In this approach, the test set is partitioned into 10 folds, each containing an equal number of objects and maintaining proportional distributions of decision classes. During each

iteration, the network is trained using 9 folds of prediction vectors and the model's performance is assessed on the remaining fold. This process is repeated three times and the results are averaged to ascertain the model's accuracy. The most important impact studied in this paper is the division of objects among local tables so as to gradually reduce the percentage of common objects appearing in local tables. Initially, all training data used in the experimental part were part of a single table. Then attributes are dispersed among the local tables. However, a small part of the attributes are common between two or more tables. Objects are divided among local tables in accordance with Algorithm 3.1. In the first step, the minimum number of objects each local table must have for there to be no common objects is calculated. The objects are then distributed in a stratified manner among local tables. A given p percentage of objects is selected from each local table. An equal fraction of this drawn set of objects is then added to all other local tables. As can be seen, we take care of the stratified distribution of objects and that only a fraction of p percent of the objects from a given local table are included in all other local decision tables. Suppose we have an original single dataset M with n objects and we want 3 local tables (T_1, T_2, T_3). Each table T_i will originally have $n//3$ objects. After which a fraction p will be selected from each T_i and shared equally among the remaining T_j local tables. This way, we would have objects common between tables (T_1, T_2), (T_1, T_3), (T_1, T_3). In the study, different percentages of common objects are checked, which are reduced by a step: 60%, 45%, 30% and 15%.

Algorithm 3.1 Algorithm of division objects between local tables.

Input: A set of objects U with cardinality N,
p - percentage of diversification, percentage of common objects,
k - number of tables.
Output: Universe of local tables, $U_i, i \in \{1, \ldots, k\}$.

numOfLocalObj $= \lfloor \frac{N}{k} \rfloor$;
remainingObj $= N - (k \cdot$ numOfLocalObj);
We divide objects from U in a stratified and disjoint manner into k subsets U_i each containing numOfLocalObj;
Objects from the set U that are not included in any local set U_i are added to all local sets U_i, their number is equal to remainingObj;
for each $i = 1 \ldots k$ do
$\rightarrow$ Choose p percent of the object from the set U_i;
$\rightarrow$ Save in S_i;
for each $i = 1 \ldots k$ do
$\rightarrow U_i = U_i \cup \bigcup_{j \in \{1,\ldots,k\}, j \neq i} \left\lfloor \frac{S_j}{(k-1)} \right\rfloor$
Return $U_i, i \in \{1, \ldots, k\}$

4. Datasets and Results

The experimental evaluation of the proposed system involves testing across five distinct datasets obtained from the UC Irvine Machine Learning Repository.[21–25] Let us delve into the specifics of each dataset.

The Vehicle Silhouettes dataset seeks to classify vehicle shapes into 4 categories by leveraging features extracted from images captured at various angles. This dataset is characterized by 18 quantitative attributes, 4 decision classes, and a total of 846 objects. In the Landsat Satellite dataset the task is to classify Earth types in satellite images based on multi-spectral pixel values within a 3×3 neighborhood. This dataset has 36 quantitative attributes, 6 decision classes, and a substantial 6435 objects. Lastly, the Dry Bean dataset focuses on the classification of bean types, employing characteristics extracted from high-resolution images subjected to segmentation and feature extraction processes. It comprises 17 quantitative attributes, 7 decision classes, and an extensive 13,611 objects. The Crowd Sourced dataset aimed at mapping geographic areas. It includes various attributes related to geographic features such as location coordinates, feature types (e.g. roads, buildings, natural features), timestamps, and possibly user metadata. This dataset has 29 quantitative attributes, 6 decision classes, and 10,546 objects. The Anuran Calls dataset is a meticulously curated collection of audio features derived from frog calls, collected from various environments. The recordings are processed to extract Mel-Frequency Cepstral Coefficients (MFCCs), which are commonly used features in audio processing and speech recognition. The dataset includes 7195 instances, each representing an audio segment of a frog call. There are 22 quantitative attributes, 4 decision classes (Family is the decision).

As part of the data pre-processing phase, a strategy of random dispersion of attributes and objects into 3, 5, 7, 9, and 11 local tables and different degree of percentage of common objects: small – 15%, medium – 30%, large – 45% and very large – 60% is examined. The method of generating dispersed decision tables is consistent with that described in the previous section and presented in Algorithm 3.1. All the datasets used are imbalanced with varying object counts across decision classes in both training and test sets. To explore the impact of imbalance on the proposed method and occurrence of common objects, two variants are considered for each dataset: experiments on dispersed imbalanced data and on dispersed balanced data modified generated by the use of the Synthetic Minority Oversampling Technique (SMOTE) method.[26]

The evaluation of classification performance is conducted on the test set, employing different measures. These measures include the Classification Accuracy (acc), which represents the fraction of correctly classified objects in the test set. Additionally, Recall evaluates the classifier's proficiency in recognizing a given class, while Precision (Prec.) expresses the classifier's precision in avoiding mistakes during classification. The F-measure (F-m.) provides a balanced assessment by combining

Precision and Recall through the formula

$$\text{F-measure } = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{3}$$

Finally, Balanced Accuracy (bacc) calculates the average Recall for all decision classes, ensuring an equitable consideration of classification accuracy across all classes. All these metrics are calculated using functions implemented in Python in the sklearn library and using weighted average option. That is, the individual metrics are counted for each decision class and their weighted average (weights are the fraction of objects in each decision class) are found. This approach is very useful as it takes into account the imbalance of decision classes.

The experiments are carried out according to the following scheme:

- Generating prediction vectors from local tables using the k-nearest neighbors classifier involving an exploration of three values for the k parameter across each dataset: $k \in \{1, 5, 10\}$;
- Generating predictions for different numbers of neurons in the hidden layer of the RBF network. The number of neurons depends on the number of neurons in the input layer of the network (which is equal to the number of decision classes times the number of local tables). The experimented values are $\{0.25, 0.5, 0.75, 1, 1.5, 1.75, 2, 2.5, 2.75, 3, 3.5, 3.75, 4, 4.5, 4.75, 5\}$ times the number of neurons in the hidden layer.

The following research questions are studied when comparing the experimental results.

- Whether the number of common objects among local tables affect the classification quality of the RBF networks;
- Whether data balancing affects the classification quality of the RBF networks;
- Whether the parameter k in the k-nearest neighbors algorithm affects the classification quality of the RBF networks used as a fusion method.

The results obtained using the RBF network as a fusion method for different values of the parameter k ($k = 1$ are Tables 1 and 2, $k = 5$ are Tables 3 and 4 and $k = 10$ are Tables 5 and 6) different number of common objects among local tables (60%, 45%, 30%, 15%) and different versions of attribute dispersion (3, 5, 7, 9, 11 local tables) are shown in Tables 1–6. In Tables 1–6, the average values of results obtained from a 10-fold cross-validation on the test set are given. This means that the RBF network is trained 10 times with 9 folds and tested on one remaining fold. In addition, each test is performed three times to ensure that the results are reliable and not distorted by the influence of randomness.

We start the comparative analysis by examining whether objects diversification (percentage of common objects between tables) has a significant impact on the results obtained using RBF networks as a fusion method. Statistical tests are

Table 1. Results of Prec., Recall, F-m., bacc and acc for the proposed approach with different degrees of diversification and $k = 1$ in the nearest neighbor algorithm. Part 1.

			Imbalanced					Balanced				
Dataset	No tab.	Common obj.	Prec.	Recall	F-m.	bacc	acc	Prec.	Recall	F-m.	bacc	acc
Vehicle		60%	0.774	0.764	0.754	0.746	0.764	0.74	0.736	0.73	0.714	0.736
		45%	0.749	0.75	0.734	0.721	0.75	0.74	0.737	0.727	0.706	0.737
	3	30%	0.772	0.758	0.748	0.736	0.758	0.775	0.756	0.745	0.732	0.756
		15%	0.788	0.78	0.769	0.762	0.78	0.747	0.734	0.721	0.715	0.734
		60%	0.793	0.77	0.765	0.746	0.77	0.63	0.73	0.668	0.684	0.73
		45%	0.781	0.775	0.766	0.73	0.775	0.642	0.742	0.68	0.695	0.742
	5	30%	0.771	0.763	0.758	0.723	0.763	0.727	0.762	0.727	0.722	0.762
		15%	0.772	0.763	0.749	0.729	0.763	0.652	0.743	0.681	0.689	0.743
		60%	0.737	0.714	0.706	0.705	0.714	0.741	0.736	0.722	0.71	0.736
		45%	0.786	0.759	0.754	0.747	0.759	0.678	0.717	0.678	0.69	0.717
	7	30%	0.773	0.749	0.745	0.728	0.749	0.761	0.742	0.733	0.718	0.742
		15%	0.736	0.733	0.723	0.711	0.733	0.71	0.714	0.698	0.685	0.714
		60%	0.7	0.701	0.684	0.67	0.701	0.721	0.721	0.708	0.685	0.721
		45%	0.788	0.775	0.755	0.74	0.775	0.635	0.715	0.66	0.681	0.715
	9	30%	0.77	0.735	0.718	0.711	0.735	0.685	0.692	0.668	0.652	0.692
		15%	0.782	0.762	0.752	0.741	0.762	0.715	0.712	0.694	0.67	0.712
		60%	0.726	0.733	0.707	0.688	0.733	0.729	0.73	0.716	0.694	0.73
		45%	0.76	0.741	0.736	0.718	0.741	0.727	0.726	0.705	0.687	0.726
	11	30%	0.751	0.739	0.733	0.709	0.739	0.738	0.735	0.722	0.712	0.735
		15%	0.775	0.754	0.741	0.724	0.754	0.773	0.765	0.747	0.74	0.765
Satellite		60%	0.892	0.889	0.887	0.872	0.889	0.893	0.89	0.887	0.862	0.89
		45%	0.906	0.904	0.902	0.889	0.904	0.899	0.9	0.897	0.874	0.9
	3	30%	0.902	0.899	0.897	0.885	0.899	0.896	0.892	0.89	0.867	0.892
		15%	0.905	0.903	0.901	0.887	0.903	0.898	0.894	0.893	0.87	0.894
		60%	0.899	0.897	0.895	0.88	0.897	0.884	0.88	0.878	0.857	0.88
		45%	0.894	0.89	0.888	0.877	0.89	0.884	0.881	0.88	0.858	0.881
	5	30%	0.896	0.894	0.893	0.876	0.894	0.883	0.88	0.879	0.855	0.88
		15%	0.891	0.887	0.886	0.874	0.887	0.878	0.877	0.874	0.85	0.877
		60%	0.89	0.888	0.886	0.87	0.888	0.886	0.882	0.88	0.858	0.882
		45%	0.889	0.887	0.885	0.867	0.887	0.882	0.881	0.877	0.854	0.881
	7	30%	0.894	0.89	0.889	0.877	0.89	0.886	0.884	0.881	0.858	0.884
		15%	0.891	0.887	0.886	0.871	0.887	0.877	0.873	0.869	0.839	0.873
		60%	0.898	0.893	0.892	0.88	0.893	0.87	0.871	0.867	0.841	0.871
		45%	0.887	0.884	0.882	0.867	0.884	0.877	0.876	0.873	0.849	0.876
	9	30%	0.895	0.889	0.888	0.877	0.889	0.87	0.871	0.868	0.84	0.871
		15%	0.895	0.891	0.89	0.876	0.891	0.861	0.862	0.853	0.819	0.862
		60%	0.888	0.886	0.884	0.87	0.886	0.884	0.88	0.879	0.858	0.88
		45%	0.886	0.882	0.881	0.866	0.882	0.885	0.883	0.88	0.85	0.883
	11	30%	0.891	0.886	0.885	0.871	0.886	0.877	0.874	0.871	0.843	0.874
		15%	0.885	0.882	0.88	0.866	0.882	0.879	0.876	0.874	0.853	0.876
Dry Bean		60%	0.918	0.917	0.917	0.928	0.917	0.92	0.918	0.918	0.927	0.918
		45%	0.92	0.919	0.919	0.928	0.919	0.916	0.914	0.914	0.925	0.914
	3	30%	0.918	0.917	0.916	0.926	0.917	0.917	0.916	0.916	0.926	0.916
		15%	0.922	0.921	0.92	0.931	0.921	0.92	0.918	0.918	0.926	0.918

Table 1. (*Continued*)

			Imbalanced					Balanced				
Dataset	No tab.	Common obj.	Prec.	Recall	F-m.	bacc	acc	Prec.	Recall	F-m.	bacc	acc
		60%	0.922	0.921	0.921	0.93	0.921	0.919	0.918	0.918	0.926	0.918
		45%	0.922	0.921	0.921	0.931	0.921	0.92	0.918	0.918	0.929	0.918
	5	30%	0.923	0.922	0.922	0.932	0.922	0.918	0.917	0.916	0.926	0.917
		15%	0.921	0.92	0.92	0.929	0.92	0.917	0.916	0.916	0.926	0.916
		60%	0.924	0.923	0.923	0.932	0.923	0.919	0.918	0.918	0.927	0.918
		45%	0.92	0.918	0.918	0.928	0.918	0.919	0.918	0.918	0.927	0.918
	7	30%	0.922	0.921	0.921	0.932	0.921	0.921	0.92	0.92	0.927	0.92
		15%	0.923	0.921	0.921	0.93	0.921	0.92	0.918	0.918	0.927	0.918
		60%	0.919	0.918	0.917	0.926	0.918	0.921	0.92	0.919	0.928	0.92
		45%	0.922	0.921	0.921	0.929	0.921	0.917	0.916	0.915	0.925	0.916
	9	30%	0.919	0.918	0.918	0.926	0.918	0.919	0.918	0.918	0.926	0.918
		15%	0.923	0.922	0.922	0.931	0.922	0.922	0.921	0.921	0.929	0.921
		60%	0.921	0.92	0.92	0.93	0.92	0.918	0.917	0.917	0.926	0.917
		45%	0.92	0.918	0.918	0.927	0.918	0.918	0.916	0.916	0.923	0.916
	11	30%	0.919	0.918	0.918	0.928	0.918	0.917	0.916	0.916	0.925	0.916
		15%	0.921	0.919	0.919	0.929	0.919	0.917	0.916	0.915	0.924	0.916

Table 2. Results of Prec., Recall, F-m., bacc and acc for the proposed approach with different degrees of diversification and $k = 1$ in the nearest neighbor algorithm. Part 2.

			Imbalanced					Balanced				
Dataset	No. tab.	Common obj.	Prec.	Recall	F-m.	bacc	acc	Prec.	Recall	F-m.	bacc	acc
Crowd		60%	0.941	0.941	0.937	0.794	0.941	0.996	0.996	0.996	0.996	0.996
Sourced		45%	0.947	0.947	0.943	0.785	0.947	0.993	0.993	0.993	0.993	0.993
	3	30%	0.945	0.943	0.94	0.803	0.943	0.996	0.996	0.996	0.996	0.996
		15%	0.949	0.949	0.945	0.81	0.949	0.997	0.997	0.997	0.997	0.997
		60%	0.921	0.925	0.92	0.736	0.925	0.989	0.989	0.989	0.988	0.989
		45%	0.925	0.931	0.924	0.73	0.931	0.987	0.987	0.987	0.987	0.987
	5	30%	0.926	0.928	0.923	0.756	0.928	0.991	0.991	0.991	0.991	0.991
		15%	0.929	0.932	0.927	0.766	0.932	0.991	0.991	0.991	0.991	0.991
		60%	0.898	0.905	0.897	0.678	0.905	0.977	0.977	0.977	0.977	0.977
		45%	0.9	0.907	0.899	0.703	0.907	0.976	0.975	0.975	0.975	0.975
	7	30%	0.902	0.908	0.901	0.703	0.908	0.977	0.976	0.976	0.976	0.976
		15%	0.893	0.901	0.892	0.671	0.901	0.978	0.978	0.978	0.978	0.978
		60%	0.901	0.911	0.901	0.666	0.911	0.967	0.966	0.966	0.967	0.966
		45%	0.911	0.917	0.908	0.696	0.917	0.96	0.96	0.96	0.96	0.96
	9	30%	0.901	0.91	0.9	0.674	0.91	0.967	0.966	0.966	0.967	0.966
		15%	0.89	0.9	0.89	0.662	0.9	0.966	0.966	0.966	0.966	0.966
		60%	0.898	0.907	0.898	0.678	0.907	0.962	0.962	0.962	0.962	0.962
		45%	0.895	0.906	0.894	0.647	0.906	0.958	0.957	0.957	0.957	0.957
	11	30%	0.892	0.906	0.894	0.663	0.906	0.963	0.962	0.962	0.962	0.962
		15%	0.902	0.912	0.901	0.665	0.912	0.96	0.959	0.959	0.959	0.959
Anuran		60%	0.993	0.993	0.993	0.987	0.993	0.994	0.994	0.994	0.994	0.994
Calls		45%	0.99	0.99	0.99	0.986	0.99	0.995	0.995	0.995	0.995	0.995
	3	30%	0.992	0.991	0.991	0.987	0.991	0.994	0.994	0.994	0.994	0.994
		15%	0.992	0.991	0.991	0.986	0.991	0.995	0.995	0.995	0.995	0.995

Table 2. (*Continued*)

			Imbalanced					Balanced				
Dataset	No. tab.	Common obj.	Prec.	Recall	F-m.	bacc	acc	Prec.	Recall	F-m.	bacc	acc
		60%	0.985	0.985	0.985	0.975	0.985	0.994	0.993	0.993	0.994	0.993
		45%	0.987	0.986	0.986	0.978	0.986	0.993	0.993	0.993	0.993	0.993
	5	30%	0.987	0.987	0.987	0.979	0.987	0.993	0.993	0.993	0.993	0.993
		15%	0.985	0.984	0.984	0.976	0.984	0.994	0.994	0.994	0.994	0.994
		60%	0.984	0.983	0.983	0.978	0.983	0.993	0.993	0.993	0.993	0.993
		45%	0.984	0.984	0.984	0.978	0.984	0.992	0.992	0.992	0.992	0.992
	7	30%	0.987	0.987	0.987	0.982	0.987	0.993	0.993	0.993	0.993	0.993
		15%	0.987	0.986	0.986	0.977	0.986	0.993	0.993	0.993	0.993	0.993
		60%	0.983	0.982	0.982	0.975	0.982	0.991	0.991	0.991	0.991	0.991
		45%	0.982	0.981	0.981	0.976	0.981	0.992	0.992	0.992	0.992	0.992
	9	30%	0.978	0.978	0.977	0.966	0.978	0.991	0.991	0.991	0.991	0.991
		15%	0.979	0.979	0.979	0.97	0.979	0.993	0.992	0.992	0.993	0.992
		60%	0.98	0.979	0.979	0.973	0.979	0.992	0.992	0.992	0.992	0.992
		45%	0.982	0.981	0.982	0.977	0.981	0.993	0.993	0.993	0.993	0.993
	11	30%	0.978	0.977	0.977	0.976	0.977	0.992	0.992	0.992	0.992	0.992
		15%	0.978	0.978	0.977	0.964	0.978	0.992	0.992	0.992	0.992	0.992

Table 3. Results of Prec., Recall, F-m., bacc and acc for the proposed approach with different degrees of diversification and $k = 5$ in the nearest neighbor algorithm.

			Imbalanced					Balanced				
Dataset	No. tab.	Common obj.	Prec.	Recall	F-m.	bacc	acc	Prec.	Recall	F-m.	bacc	acc
Vehicle		60%	0.745	0.734	0.701	0.705	0.734	0.766	0.748	0.742	0.729	0.748
		45%	0.776	0.761	0.749	0.734	0.761	0.748	0.727	0.715	0.701	0.727
	3	30%	0.776	0.748	0.74	0.713	0.748	0.741	0.742	0.727	0.726	0.742
		15%	0.799	0.782	0.774	0.759	0.782	0.774	0.765	0.751	0.743	0.765
		60%	0.734	0.728	0.716	0.7	0.728	0.721	0.72	0.698	0.68	0.72
		45%	0.794	0.777	0.769	0.752	0.777	0.781	0.776	0.763	0.739	0.776
	5	30%	0.773	0.765	0.752	0.723	0.765	0.764	0.768	0.754	0.737	0.768
		15%	0.772	0.753	0.746	0.718	0.753	0.8	0.785	0.779	0.758	0.785
		60%	0.737	0.734	0.72	0.714	0.734	0.776	0.745	0.735	0.729	0.745
		45%	0.774	0.767	0.759	0.742	0.767	0.746	0.738	0.726	0.715	0.738
	7	30%	0.777	0.761	0.747	0.733	0.761	0.74	0.728	0.713	0.696	0.728
		15%	0.782	0.773	0.765	0.742	0.773	0.747	0.748	0.727	0.723	0.748
		60%	0.762	0.74	0.734	0.715	0.74	0.76	0.735	0.73	0.71	0.735
		45%	0.763	0.743	0.737	0.731	0.743	0.717	0.734	0.711	0.704	0.734
	9	30%	0.755	0.744	0.74	0.726	0.744	0.721	0.732	0.706	0.705	0.732
		15%	0.767	0.757	0.749	0.743	0.757	0.719	0.708	0.696	0.674	0.708
		60%	0.766	0.764	0.758	0.733	0.764	0.753	0.743	0.731	0.722	0.743
		45%	0.777	0.757	0.743	0.721	0.757	0.75	0.75	0.734	0.721	0.75
	11	30%	0.737	0.748	0.735	0.72	0.748	0.736	0.722	0.715	0.698	0.722
		15%	0.8	0.783	0.769	0.751	0.783	0.752	0.749	0.737	0.728	0.749
Satellite		60%	0.89	0.887	0.885	0.87	0.887	0.894	0.896	0.892	0.866	0.896
		45%	0.898	0.896	0.894	0.879	0.896	0.902	0.898	0.896	0.872	0.898

Table 3. (*Continued*)

			Imbalanced					Balanced				
Dataset	No. tab.	Common obj.	Prec.	Recall	F-m.	bacc	acc	Prec.	Recall	F-m.	bacc	acc
	3	30%	0.895	0.893	0.891	0.874	0.893	0.893	0.892	0.89	0.867	0.892
		15%	0.902	0.9	0.899	0.884	0.9	0.903	0.901	0.899	0.875	0.901
		60%	0.892	0.889	0.887	0.871	0.889	0.896	0.891	0.889	0.869	0.891
		45%	0.9	0.897	0.895	0.882	0.897	0.889	0.888	0.885	0.862	0.888
	5	30%	0.903	0.901	0.899	0.884	0.901	0.89	0.887	0.884	0.863	0.887
		15%	0.899	0.896	0.895	0.877	0.896	0.892	0.887	0.885	0.865	0.887
		60%	0.892	0.889	0.887	0.869	0.889	0.89	0.884	0.881	0.861	0.884
		45%	0.895	0.891	0.891	0.874	0.891	0.896	0.892	0.89	0.869	0.892
	7	30%	0.895	0.891	0.89	0.88	0.891	0.884	0.883	0.88	0.856	0.883
		15%	0.898	0.894	0.893	0.88	0.894	0.888	0.886	0.883	0.859	0.886
		60%	0.898	0.894	0.893	0.878	0.894	0.877	0.876	0.873	0.851	0.876
		45%	0.9	0.896	0.894	0.882	0.896	0.884	0.884	0.881	0.858	0.884
	9	30%	0.897	0.893	0.892	0.878	0.893	0.889	0.887	0.884	0.862	0.887
		15%	0.901	0.897	0.896	0.882	0.897	0.883	0.879	0.877	0.853	0.879
		60%	0.891	0.887	0.885	0.871	0.887	0.883	0.879	0.877	0.857	0.879
		45%	0.893	0.888	0.887	0.872	0.888	0.882	0.881	0.879	0.854	0.881
	11	30%	0.893	0.891	0.889	0.873	0.891	0.877	0.875	0.873	0.849	0.875
		15%	0.894	0.89	0.889	0.876	0.89	0.888	0.884	0.882	0.862	0.884
Dry Bean		60%	0.921	0.92	0.92	0.931	0.92	0.922	0.921	0.92	0.931	0.921
		45%	0.921	0.921	0.92	0.931	0.921	0.921	0.92	0.92	0.929	0.92
	3	30%	0.92	0.92	0.919	0.93	0.92	0.921	0.92	0.919	0.927	0.92
		15%	0.922	0.921	0.92	0.931	0.921	0.919	0.918	0.918	0.928	0.918
		60%	0.923	0.922	0.922	0.931	0.922	0.919	0.919	0.918	0.927	0.919
		45%	0.922	0.92	0.92	0.932	0.92	0.918	0.917	0.917	0.926	0.917
	5	30%	0.923	0.922	0.922	0.932	0.922	0.92	0.919	0.919	0.927	0.919
		15%	0.921	0.92	0.919	0.93	0.92	0.92	0.919	0.919	0.928	0.919
		60%	0.922	0.921	0.92	0.931	0.921	0.92	0.919	0.918	0.928	0.919
		45%	0.921	0.92	0.92	0.93	0.92	0.922	0.921	0.921	0.929	0.921
	7	30%	0.92	0.919	0.919	0.929	0.919	0.921	0.92	0.92	0.93	0.92
		15%	0.921	0.92	0.919	0.931	0.92	0.921	0.92	0.92	0.93	0.92
		60%	0.921	0.92	0.92	0.928	0.92	0.921	0.92	0.92	0.928	0.92
		45%	0.92	0.918	0.918	0.928	0.918	0.92	0.918	0.918	0.928	0.918
	9	30%	0.921	0.92	0.92	0.929	0.92	0.922	0.92	0.92	0.928	0.92
		15%	0.921	0.92	0.92	0.93	0.92	0.921	0.92	0.92	0.928	0.92
		60%	0.919	0.918	0.918	0.928	0.918	0.919	0.918	0.918	0.926	0.918
		45%	0.919	0.918	0.918	0.928	0.918	0.919	0.918	0.918	0.926	0.918
	11	30%	0.921	0.92	0.919	0.929	0.92	0.919	0.917	0.917	0.925	0.917
		15%	0.922	0.921	0.92	0.93	0.921	0.918	0.917	0.916	0.926	0.917

Table 4. Results of Prec., Recall, F-m., bacc and acc for the proposed approach with different degrees of diversification and $k = 5$ in the nearest neighbor algorithm. Part 2.

			Imbalanced					Balanced				
Dataset	No. tab.	Common obj.	Prec.	Recall	F-m.	bacc	acc	Prec.	Recall	F-m.	bacc	acc
Crowd		60%	0.946	0.944	0.941	0.803	0.944	0.992	0.992	0.992	0.992	0.992
Sourced		45%	0.947	0.946	0.943	0.801	0.946	0.988	0.988	0.988	0.988	0.988
	3	30%	0.937	0.94	0.936	0.773	0.94	0.993	0.993	0.993	0.993	0.993

Table 4. (*Continued*)

			Imbalanced					Balanced				
Dataset	No. tab.	Common obj.	Prec.	Recall	F-m.	bacc	acc	Prec.	Recall	F-m.	bacc	acc
		15%	0.946	0.944	0.941	0.819	0.944	0.99	0.99	0.99	0.99	0.99
		60%	0.923	0.928	0.921	0.734	0.928	0.981	0.981	0.981	0.981	0.981
		45%	0.932	0.935	0.929	0.757	0.935	0.977	0.976	0.976	0.976	0.976
	5	30%	0.932	0.935	0.929	0.76	0.935	0.981	0.981	0.981	0.981	0.981
		15%	0.934	0.935	0.931	0.798	0.935	0.978	0.978	0.978	0.978	0.978
		60%	0.91	0.918	0.91	0.704	0.918	0.97	0.97	0.97	0.97	0.97
		45%	0.906	0.913	0.904	0.684	0.913	0.964	0.963	0.963	0.963	0.963
	7	30%	0.91	0.916	0.909	0.708	0.916	0.968	0.968	0.968	0.968	0.968
		15%	0.906	0.915	0.906	0.697	0.915	0.963	0.962	0.962	0.962	0.962
		60%	0.907	0.916	0.906	0.684	0.916	0.96	0.959	0.959	0.96	0.959
		45%	0.908	0.915	0.906	0.671	0.915	0.949	0.948	0.948	0.948	0.948
	9	30%	0.904	0.915	0.904	0.662	0.915	0.959	0.958	0.958	0.958	0.958
		15%	0.91	0.916	0.908	0.698	0.916	0.949	0.947	0.947	0.948	0.947
		60%	0.906	0.913	0.904	0.699	0.913	0.958	0.957	0.957	0.957	0.957
		45%	0.904	0.912	0.903	0.684	0.912	0.945	0.944	0.944	0.944	0.944
	11	30%	0.846	0.872	0.85	0.548	0.872	0.959	0.957	0.957	0.957	0.957
		15%	0.903	0.912	0.902	0.68	0.912	0.946	0.944	0.944	0.944	0.944
Anuran Calls		60%	0.99	0.99	0.99	0.984	0.99	0.993	0.993	0.993	0.993	0.993
		45%	0.99	0.99	0.99	0.979	0.99	0.992	0.992	0.992	0.992	0.992
	3	30%	0.989	0.988	0.989	0.981	0.988	0.993	0.993	0.993	0.993	0.993
		15%	0.989	0.988	0.988	0.982	0.988	0.993	0.993	0.993	0.993	0.993
		60%	0.986	0.985	0.985	0.981	0.985	0.993	0.993	0.993	0.993	0.993
		45%	0.985	0.984	0.984	0.976	0.984	0.993	0.993	0.993	0.993	0.993
	5	30%	0.983	0.983	0.983	0.975	0.983	0.992	0.992	0.992	0.992	0.992
		15%	0.985	0.984	0.984	0.976	0.984	0.992	0.992	0.992	0.992	0.992
		60%	0.984	0.983	0.983	0.974	0.983	0.992	0.992	0.992	0.992	0.992
		45%	0.982	0.982	0.982	0.976	0.982	0.991	0.991	0.991	0.991	0.991
	7	30%	0.983	0.983	0.983	0.977	0.983	0.992	0.992	0.992	0.992	0.992
		15%	0.984	0.984	0.983	0.978	0.984	0.992	0.992	0.992	0.992	0.992
		60%	0.984	0.983	0.983	0.977	0.983	0.991	0.991	0.991	0.991	0.991
		45%	0.98	0.98	0.98	0.975	0.98	0.993	0.993	0.993	0.993	0.993
	9	30%	0.984	0.984	0.984	0.975	0.984	0.991	0.991	0.991	0.991	0.991
		15%	0.978	0.977	0.977	0.968	0.977	0.991	0.991	0.991	0.992	0.991
		60%	0.978	0.978	0.978	0.965	0.978	0.992	0.992	0.992	0.992	0.992
		45%	0.98	0.979	0.979	0.976	0.979	0.99	0.99	0.99	0.99	0.99
	11	30%	0.978	0.977	0.977	0.971	0.977	0.992	0.992	0.992	0.992	0.992
		15%	0.978	0.977	0.977	0.972	0.977	0.991	0.991	0.991	0.991	0.991

Table 5. Results of Prec., Recall, F-m., bacc and acc for the proposed approach with different degrees of diversification and $k = 10$ in the nearest neighbor algorithm.

			Imbalanced					Balanced				
Dataset	No. tab.	Common obj.	Prec.	Recall	F-m.	bacc	acc	Prec.	Recall	F-m.	bacc	acc
Vehicle		60%	0.753	0.736	0.726	0.709	0.736	0.754	0.752	0.741	0.737	0.752
		45%	0.774	0.751	0.735	0.713	0.751	0.746	0.747	0.734	0.725	0.747
	3	30%	0.776	0.741	0.731	0.715	0.741	0.746	0.736	0.717	0.71	0.736

Table 5. (*Continued*)

			Imbalanced					Balanced				
Dataset	No. tab.	Common obj.	Prec.	Recall	F-m.	bacc	acc	Prec.	Recall	F-m.	bacc	acc
		15%	0.744	0.74	0.734	0.707	0.74	0.768	0.767	0.756	0.743	0.767
		60%	0.721	0.717	0.704	0.682	0.717	0.772	0.757	0.749	0.724	0.757
		45%	0.802	0.784	0.775	0.749	0.784	0.782	0.777	0.763	0.746	0.777
	5	30%	0.765	0.756	0.743	0.722	0.756	0.786	0.771	0.766	0.749	0.771
		15%	0.758	0.738	0.726	0.705	0.738	0.777	0.762	0.756	0.736	0.762
		60%	0.756	0.735	0.725	0.718	0.735	0.744	0.737	0.725	0.712	0.737
		45%	0.792	0.782	0.778	0.758	0.782	0.763	0.755	0.741	0.723	0.755
	7	30%	0.744	0.75	0.73	0.709	0.75	0.758	0.745	0.733	0.718	0.745
		15%	0.763	0.754	0.743	0.721	0.754	0.737	0.743	0.724	0.717	0.743
		60%	0.768	0.742	0.733	0.716	0.742	0.728	0.716	0.703	0.692	0.716
		45%	0.762	0.753	0.747	0.736	0.753	0.737	0.732	0.714	0.705	0.732
	9	30%	0.753	0.741	0.733	0.719	0.741	0.726	0.735	0.714	0.708	0.735
		15%	0.76	0.755	0.746	0.731	0.755	0.712	0.711	0.694	0.683	0.711
		60%	0.742	0.741	0.732	0.71	0.741	0.761	0.747	0.739	0.717	0.747
		45%	0.779	0.772	0.757	0.734	0.772	0.737	0.743	0.722	0.704	0.743
	11	30%	0.759	0.753	0.745	0.725	0.753	0.724	0.729	0.714	0.692	0.729
		15%	0.765	0.764	0.749	0.73	0.764	0.755	0.748	0.738	0.72	0.748
Satellite		60%	0.896	0.892	0.89	0.874	0.892	0.898	0.897	0.894	0.872	0.897
		45%	0.895	0.892	0.89	0.875	0.892	0.895	0.893	0.891	0.869	0.893
	3	30%	0.896	0.893	0.892	0.877	0.893	0.898	0.897	0.894	0.872	0.897
		15%	0.903	0.901	0.899	0.881	0.901	0.896	0.895	0.893	0.87	0.895
		60%	0.891	0.886	0.885	0.867	0.886	0.89	0.886	0.884	0.864	0.886
		45%	0.896	0.893	0.892	0.877	0.893	0.889	0.886	0.884	0.861	0.886
	5	30%	0.9	0.896	0.895	0.881	0.896	0.893	0.89	0.888	0.867	0.89
		15%	0.895	0.891	0.89	0.875	0.891	0.895	0.892	0.89	0.87	0.892
		60%	0.894	0.89	0.889	0.873	0.89	0.888	0.882	0.88	0.861	0.882
		45%	0.896	0.89	0.89	0.878	0.89	0.893	0.891	0.889	0.867	0.891
	7	30%	0.897	0.893	0.892	0.878	0.893	0.887	0.885	0.883	0.86	0.885
		15%	0.897	0.892	0.891	0.878	0.892	0.889	0.885	0.882	0.86	0.885
		60%	0.893	0.888	0.886	0.87	0.888	0.885	0.881	0.879	0.857	0.881
		45%	0.894	0.891	0.89	0.878	0.891	0.892	0.889	0.887	0.867	0.889
	9	30%	0.899	0.895	0.894	0.879	0.895	0.885	0.881	0.879	0.857	0.881
		15%	0.897	0.894	0.892	0.876	0.894	0.887	0.883	0.88	0.858	0.883
		60%	0.888	0.884	0.882	0.868	0.884	0.883	0.879	0.877	0.856	0.879
		45%	0.896	0.892	0.891	0.875	0.892	0.883	0.883	0.879	0.855	0.883
	11	30%	0.896	0.892	0.891	0.878	0.892	0.878	0.877	0.873	0.85	0.877
		15%	0.891	0.888	0.886	0.871	0.888	0.885	0.88	0.877	0.859	0.88
Dry Bean		60%	0.919	0.918	0.917	0.927	0.918	0.919	0.918	0.918	0.928	0.918
		45%	0.921	0.92	0.92	0.931	0.92	0.921	0.92	0.92	0.929	0.92
	3	30%	0.923	0.922	0.922	0.931	0.922	0.922	0.921	0.921	0.931	0.921
		15%	0.922	0.921	0.921	0.931	0.921	0.918	0.917	0.917	0.926	0.917
		60%	0.922	0.921	0.921	0.931	0.921	0.92	0.919	0.919	0.927	0.919
		45%	0.923	0.922	0.922	0.931	0.922	0.92	0.919	0.918	0.928	0.919
	5	30%	0.922	0.921	0.921	0.931	0.921	0.921	0.919	0.919	0.927	0.919
		15%	0.923	0.922	0.922	0.932	0.922	0.921	0.92	0.919	0.928	0.92
		60%	0.921	0.92	0.92	0.93	0.92	0.919	0.918	0.918	0.928	0.918
		45%	0.922	0.92	0.92	0.931	0.92	0.918	0.917	0.917	0.927	0.917

Table 5. (*Continued*)

			Imbalanced					Balanced				
Dataset	No. tab.	Common obj.	Prec.	Recall	F-m.	bacc	acc	Prec.	Recall	F-m.	bacc	acc
	7	30%	0.92	0.919	0.919	0.927	0.919	0.92	0.919	0.918	0.927	0.919
		15%	0.92	0.919	0.919	0.93	0.919	0.919	0.919	0.918	0.927	0.919
		60%	0.92	0.919	0.919	0.927	0.919	0.921	0.919	0.919	0.928	0.919
		45%	0.919	0.918	0.918	0.927	0.918	0.919	0.918	0.918	0.927	0.918
	9	30%	0.92	0.919	0.919	0.928	0.919	0.918	0.917	0.917	0.927	0.917
		15%	0.92	0.919	0.919	0.928	0.919	0.921	0.919	0.919	0.928	0.919
		60%	0.919	0.918	0.918	0.928	0.918	0.919	0.917	0.917	0.926	0.917
		45%	0.921	0.919	0.919	0.928	0.919	0.92	0.918	0.918	0.928	0.918
	11	30%	0.919	0.918	0.918	0.927	0.918	0.92	0.919	0.919	0.927	0.919
		15%	0.923	0.922	0.922	0.932	0.922	0.92	0.919	0.919	0.929	0.919

Table 6. Results of Prec., Recall, F-m., bacc and acc for the proposed approach with different degrees of diversification and $k = 10$ in the nearest neighbor algorithm. Part 2.

			Imbalanced					Balanced				
Dataset	No. tab.	Common obj.	Prec.	Recall	F-m.	bacc	acc	Prec.	Recall	F-m.	bacc	acc
Crowd		60%	0.939	0.937	0.934	0.786	0.937	0.985	0.985	0.985	0.985	0.985
Sourced		45%	0.938	0.939	0.935	0.785	0.939	0.983	0.982	0.982	0.983	0.982
	3	30%	0.934	0.934	0.929	0.761	0.934	0.986	0.986	0.986	0.986	0.986
		15%	0.939	0.94	0.936	0.781	0.94	0.982	0.982	0.982	0.982	0.982
		60%	0.924	0.929	0.922	0.73	0.929	0.973	0.972	0.972	0.972	0.972
		45%	0.924	0.931	0.922	0.726	0.931	0.97	0.969	0.969	0.969	0.969
	5	30%	0.929	0.937	0.93	0.744	0.937	0.97	0.97	0.97	0.97	0.97
		15%	0.933	0.937	0.932	0.775	0.937	0.969	0.968	0.968	0.969	0.968
		60%	0.91	0.914	0.907	0.687	0.914	0.963	0.963	0.963	0.963	0.963
		45%	0.915	0.919	0.912	0.713	0.919	0.956	0.955	0.955	0.955	0.955
	7	30%	0.915	0.921	0.914	0.701	0.921	0.963	0.962	0.962	0.962	0.962
		15%	0.912	0.917	0.91	0.694	0.917	0.953	0.952	0.952	0.951	0.952
		60%	0.907	0.915	0.906	0.669	0.915	0.957	0.957	0.957	0.957	0.957
		45%	0.907	0.914	0.906	0.667	0.914	0.942	0.941	0.941	0.941	0.941
	9	30%	0.906	0.914	0.905	0.672	0.914	0.957	0.956	0.956	0.956	0.956
		15%	0.909	0.915	0.907	0.685	0.915	0.941	0.939	0.939	0.939	0.939
		60%	0.903	0.912	0.903	0.697	0.912	0.957	0.955	0.956	0.955	0.955
		45%	0.906	0.913	0.904	0.69	0.913	0.939	0.937	0.937	0.937	0.937
	11	30%	0.897	0.908	0.897	0.677	0.908	0.954	0.953	0.953	0.953	0.953
		15%	0.9	0.91	0.9	0.676	0.91	0.939	0.937	0.937	0.937	0.937
Anuran		60%	0.986	0.986	0.986	0.98	0.986	0.991	0.991	0.991	0.991	0.991
Calls		45%	0.988	0.987	0.987	0.982	0.987	0.991	0.991	0.991	0.991	0.991
	3	30%	0.983	0.984	0.983	0.964	0.984	0.991	0.991	0.991	0.991	0.991
		15%	0.987	0.987	0.987	0.98	0.987	0.992	0.992	0.992	0.992	0.992
		60%	0.981	0.981	0.98	0.974	0.981	0.992	0.991	0.991	0.992	0.991
		45%	0.979	0.979	0.978	0.969	0.979	0.991	0.991	0.991	0.991	0.991
	5	30%	0.978	0.978	0.978	0.973	0.978	0.991	0.991	0.991	0.991	0.991
		15%	0.981	0.981	0.981	0.974	0.981	0.991	0.991	0.991	0.991	0.991
		60%	0.979	0.978	0.978	0.973	0.978	0.99	0.99	0.99	0.99	0.99
		45%	0.978	0.977	0.977	0.971	0.977	0.99	0.99	0.99	0.991	0.99

Table 6. (*Continued*)

			Imbalanced					Balanced				
Dataset	No. tab.	Common obj.	Prec.	Recall	F-m.	bacc	acc	Prec.	Recall	F-m.	bacc	acc
	7	30%	0.98	0.98	0.98	0.974	0.98	0.99	0.99	0.99	0.99	0.99
		15%	0.981	0.981	0.981	0.976	0.981	0.99	0.99	0.99	0.99	0.99
		60%	0.979	0.978	0.978	0.969	0.978	0.99	0.99	0.99	0.99	0.99
		45%	0.975	0.974	0.974	0.971	0.974	0.992	0.992	0.992	0.992	0.992
	9	30%	0.982	0.981	0.981	0.971	0.981	0.991	0.991	0.991	0.991	0.991
		15%	0.979	0.978	0.978	0.974	0.978	0.991	0.991	0.991	0.991	0.991
		60%	0.977	0.976	0.976	0.969	0.976	0.991	0.991	0.991	0.991	0.991
		45%	0.979	0.978	0.978	0.972	0.978	0.99	0.99	0.99	0.99	0.99
	11	30%	0.975	0.975	0.975	0.966	0.975	0.99	0.99	0.99	0.99	0.99
		15%	0.976	0.975	0.975	0.966	0.975	0.991	0.99	0.99	0.991	0.99

performed for the three main measures: F-measure, balanced accuracy and accuracy. For each measure four dependent samples (with 60%, 45%, 30% and 15% of common objects among local tables) of 150 observations is used. The null hypothesis, H_θ is defined, whereby H_θ means there is no significance difference in measures (F-measure, balanced accuracy or accuracy) for objects diversification in size 60%, 45%, 30% and 15% in the considered model for dispersed data. The Friedman test confirmed that there is no statistical difference for any of the measures — therefore, we cannot reject the null hypothesis for no measure. More precisely, we obtain the following values of statistics and p-values: for F-measure $\chi^2(150, 3) = 2.677$, $p = 0.44$; for balanced accuracy $\chi^2(150, 3) = 7.633$, $p = 0.054$ and for accuracy $\chi^2(150, 3) = 3.774, p = 0.29$. Also the post-hoc Dunn Bonferroni test did not confirm a significant difference $p = 0.000001$. In addition, the Wilcoxon of each pair test is performed to ensure that there is no significant statistical difference between any pair of dependent samples (different objects diversification). This test confirmed that only the extreme diversification values of 15% and 60% possesses significance in the measure values. The following p-values are obtained: for F-measure $p = 0.04$; for balanced accuracy $p = 0.02$ and for accuracy $p = 0.01$. Comparative box-plot for the F-measure, balanced accuracy and accuracy with four different percentages of common objects is created (Fig. 1). As can be seen from the figure, the difference between the averages of measures is really small. So, it can be said that there is no difference in the results generated by the RBF networks as a fusion method for different degrees of common objects among the local tables. Thus, the method is resistant to the variation of objects in local tables. Ultimately, the data can be fragmented in terms of both objects and attributes. This conclusion shows the usefulness of the method even when there is a large dispersion of objects between local tables, which gives a very good prognosis for the application of the considered method.

We now examine the effect of balancing on the results obtained in terms of F-measure, balanced accuracy and accuracy. For statistical test, the two dependent

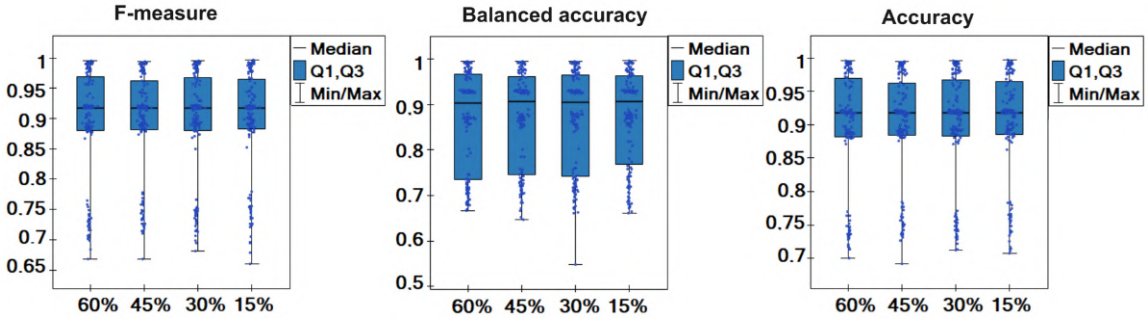


Fig. 1. Comparison of F-measure, balanced accuracy and accuracy obtained for the RBF networks as a fusion method for different degrees percentages of common objects among local tables 60%, 45%, 30% and 15%.

groups are created (imbalanced and balanced), each with 300 samples. The null hypothesis H_θ means there is no significance in F-measure, balanced accuracy or accuracy in terms of imbalanced and balanced datasets for the considered model. The Wilcoxon test for dependent samples confirmed the statistical significance of the differences for all three measures: F-measure p-value $= 0.001$, balanced accuracy p-value $= 0.0002$ and accuracy p-value $= 0.0002$. Comparative box-plot for the F-measure, balanced accuracy and accuracy for imbalanced and balanced results is created (Fig. 2). The graph also shows a significant difference in results. Definitely for balanced datasets we get better results, higher values of F-measure, balanced accuracy and accuracy. From a practical point of view, this means that before applying the proposed method, it is worth balancing the dataset (here the SMOTE method was used for this purpose).

Statistical analysis is performed to compare both the results in terms of the percentage of common objects in local tables and the imbalance of datasets. This time, eight dependent samples are compared: 60%, 45%, 30%, 15% of common objects for imbalanced datasets and 60%, 45%, 30%, 15% of common objects for balanced datasets. In this way, we overcome the situation where data balance could disturb the overall conclusion regarding the effect of the number of common objects in the dataset on the results. The null hypothesis means there is no significance difference in F-measure, balanced accuracy or accuracy in terms of imbalanced and

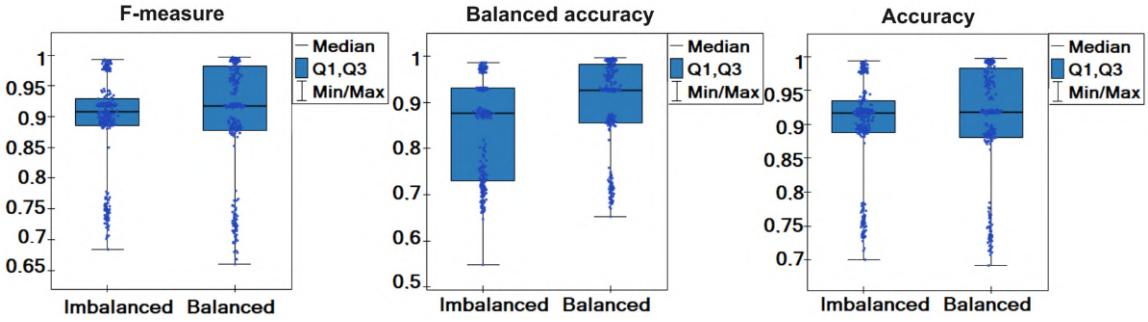


Fig. 2. Comparison of F-measure, balanced accuracy and accuracy obtained for the RBF networks as a fusion method for imbalanced and balanced datasets.

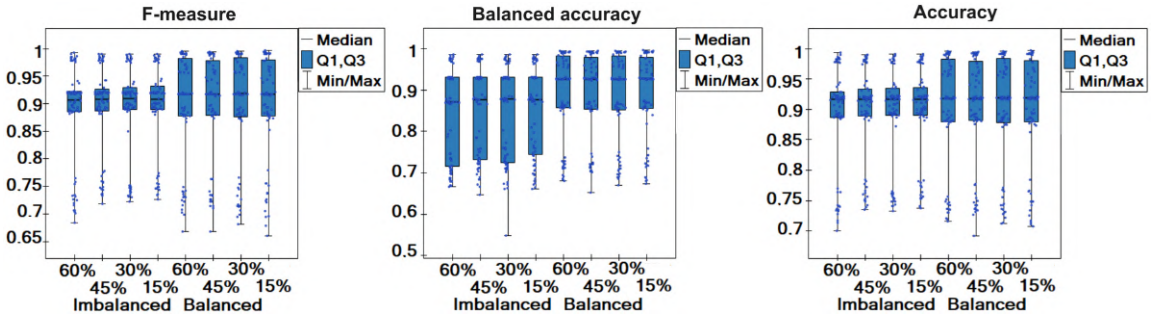


Fig. 3. Comparison of F-measure, balanced accuracy and accuracy obtained for the RBF networks as a fusion method for imbalanced and 60%, 45%, 30%, 15% of common objects among local tables and balanced datasets and 60%, 45%, 30%, 15% of common objects among local tables.

balanced datasets and different objects diversification for considered model. We are dealing with multiple dependent samples, from a ratio scale. Normality of distribution is checked with the Shapiro–Wilk test. The test confirms the non-normality of the distribution as the null hypothesis of the Shapiro–Wilk test is rejected. So, the Friedman test is chosen to compare the significance of differences. The results of this test did not allow the rejection of the null hypothesis — so there is no significant difference in the results for either measure. The individual results are as follows: for F-measure $\chi^2(75, 7) = 12.97$, $p = 0.07$; for balanced accuracy $\chi^2(75, 7) = 11.61$, $p = 0.11$ and for accuracy $\chi^2(75, 7) = 12.71$, $p = 0.08$. Comparative box-plot for the F-measure, balanced accuracy and accuracy for imbalanced and balanced datasets and 60%, 45%, 30%, 15% of common objects among local tables (Fig. 3) confirms that only balancing of datasets affects results. As can be seen, differences in the boxes can only be seen between the first four groups (for imbalanced data) and the next four groups (for balanced data). The considered method is completely insensitive to the number of common objects present in local tables.

A statistical analysis is conducted to confirm that the values of the parameter k used in the k-nearest neighbor algorithm significantly affect the F-measure, balanced accuracy, and accuracy. To achieve this, three dependent samples, each containing 200 observations, are created based on the results obtained for different values of the k parameter (separately for each measure). Since the Shapiro–Wilk test indicates that the distribution is not normal, the Friedman test is employed to verify the hypothesis of significant differences in the measure values. The Friedman test confirms the significance of these differences for all three measures, yielding the following results: for F-measure $\chi^2(200, 2) = 20.15$, $p = 0.00004$; for balanced accuracy $\chi^2(200, 2) = 21.76$, $p = 0.00002$ and for accuracy $\chi^2(200, 2) = 18.78$, $p = 0.00008$. Also the post-hoc Dunn Bonferroni test confirms significant difference between results for all values of the parameter k (except the pair $k = 1$ and $k = 10$) for the analyzed measures. Comparative box-plot for the F-measure, balanced accuracy and accuracy for different values of parameter k (Fig. 4) confirms that the value of the parameter k has a significant impact on the results obtained. However, it is not possible to unequivocally determine which value of the parameter k generates the

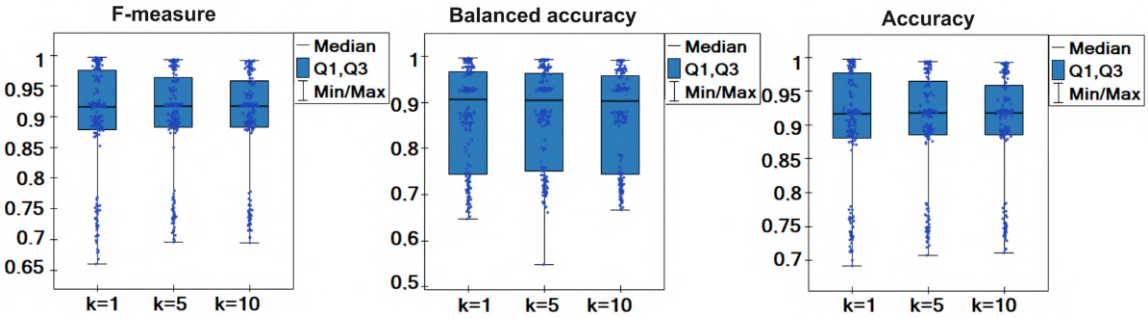


Fig. 4. Comparison of F-measure, balanced accuracy and accuracy obtained for the RBF networks as a fusion method for different values of the parameter k used in the k-nearest neighbor algorithm ($k = 1, k = 5, k = 10$).

best results as this is highly dependent on the specifics of the dataset and should always be preceded by a thorough analysis.

In summary, the most important conclusions of the above analyses are as follows:

- No significant differences are found in the performance measures (F-measure, balanced accuracy, and accuracy) when varying the percentage of common objects among local tables using RBF networks. The method is resistant to variations in the number of common objects among local tables.
- Balancing the data significantly improves the performance measures (F-measure, balanced accuracy, and accuracy). Balanced datasets yields better results compared to imbalanced ones, highlighting the importance of data balancing before applying the method.
- The values of the parameter k significantly affect the performance measures of the proposed method. It is crucial to perform a thorough analysis to determine the optimal value of k for specific datasets as the best value depends on the characteristics of the dataset.

5. Conclusions

This paper studied a classification model based on dispersed data using the k-nearest neighbor classifier as the base classifier and the RBF network as the fusion method. It was investigated whether the dispersion of both objects and attributes significantly affects the classification quality of the method. The study was conducted by gradually reducing the percentage of common attributes in local tables. The key conclusion of the study is that the proposed classification model is robust to a decreasing number of common objects in local tables. There is no significant effect on the average values of the measures like the F-measure, balanced accuracy and accuracy obtained for different percentages of common objects in local tables. In addition, it was noted that a key factor affecting the result is the balancing of the dataset, and here it came out that for dispersed data the proposed approach performs better for balanced datasets. The value of the parameter k in the nearest neighbor algorithm

was found to significantly impact the performance measures. However, determining the optimal value of requires a thorough analysis specific to the characteristics of each dataset.

Naturally, there are some limitations to the research presented in the paper. The classification model performs notably better on balanced datasets compared to imbalanced ones. As a result, classification quality may suffer in cases where data balancing techniques, such as SMOTE, cannot be effectively applied. Additionally, the model's performance may be impacted by noise, particularly in the form of variations in data intensity or attribute inconsistencies, though the paper does not examine how the model handles noisy datasets. This will be addressed in future research. Furthermore, the model's performance is sensitive to the tuning of specific parameters, including the number of neurons in the hidden layer of the RBF network and the value of k in the k-nearest neighbors classifier. Achieving optimal results, therefore, necessitates thorough parameter analysis for each dataset.

The main implication of the research is to extend the understanding of classification models in dispersed data environments. It provides new insights into applying k-nearest neighbors and RBF networks for data classification when objects and attributes are distributed across different datasets. This advancement adds to the existing literature on decentralized and ensemble learning techniques. The study demonstrates that the proposed model is robust against reducing common objects across datasets, contributing to the theory that effective classification is possible even in highly dispersed data scenarios. The proposed model can be directly applied in industries like healthcare, finance, and surveillance, where data is often dispersed across multiple sources. For example, hospitals could use this model to classify patient records across different locations without centralizing data. The findings suggest that organizations can scale their data–driven decision–making processes across multiple decentralized data repositories without losing classification accuracy.

In future work, we plan to explore the integration of the proposed model with federated learning frameworks to enhance privacy and security when dealing with sensitive and dispersed data across different locations.

ORCID

Kwabena Frimpong Marfo https://orcid.org/0000-0003-2226-9097
Małgorzata Przybyła-Kasperek https://orcid.org/0000-0003-0616-9694

References

1. K. F. Marfo and M. Przybyła-Kasperek, Exploring the impact of object diversity on classification quality in dispersed data environments, in *Asian Conf. Intelligent Information and Database Systems* (Springer Nature Singapore, Singapore, 2024), pp. 250–262.
2. Y. Song, P. N. Suganthan, W. Pedrycz, J. Ou, Y. He, Y. Chen and Y. Wu, Ensemble reinforcement learning: A survey, *Appl. Soft Comput.* **149** (2023) 110975.

3. L. Korycki and B. Krawczyk, Adversarial concept drift detection under poisoning attacks for robust data stream mining, *Mach. Learn.* **112**(10) (2023) 4013–4048.
4. M. Woźniak, P. Zyblewski and P. Ksieniewicz, Active weighted aging ensemble for drifted data stream classification, *Inf. Sci.* **630** (2023) 286–304.
5. D. Tiwari, B. Nagpal, B. S. Bhati, A. Mishra and M. Kumar, A systematic review of social network sentiment analysis with comparative study of ensemble-based techniques, *Artif. Intell. Rev.* **56**(11) (2023) 13407–13461.
6. C. Vuppalapati, *Machine Learning and Artificial Intelligence for Agricultural Economics: Prognostic Data Analytics to Serve Small Scale Farmers Worldwide*, Vol. 314 (Springer Nature, 2021).
7. J. Grzyb and M. Woźniak, SVM ensemble training for imbalanced data classification using multi-objective optimization techniques, *Appl. Intell.* **53**(12) (2023) 15424–15441.
8. J. Li, P. Liu, L. Chen, W. Pedrycz and W. Ding, An integrated fusion framework for ensemble learning leveraging gradient boosting and fuzzy rule-based models, *IEEE Trans. Artif. Intell.* **1** (2024) 1–15.
9. J. Musaev, A. Anorboev, Y. S. Seo, N. T. Nguyen and D. Hwang, ICNN-Ensemble: An improved convolutional neural network ensemble model for medical image classification, *IEEE Access* **11** (2023) 86285–86296.
10. R. Burduk and J. Biedrzycki, Subspace-based decision trees integration, *Inf. Sci.* **592** (2022) 215–226.
11. P. Trajdos, R. Burduk and A. Kasprzak, Combining linear classifiers using score function based on distance to decision boundary, in *Int. Conf. Artificial Intelligence and Soft Computing* (Springer Nature Switzerland, Cham, 2023), pp. 402–411.
12. I. Czarnowski, Agent-based population learning algorithm for over-sampling in the classification of imbalanced data streams, *Procedia Comput. Sci.* **225** (2023) 686–692.
13. R. Yang and S. Sarkar, Gesture recognition using hidden markov models from fragmented observations, in *2006 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 1 (IEEE, 2006), pp. 766–773.
14. J. Chaki and N. Dey, Fragmented handwritten digit recognition using grading scheme and fuzzy rules, *Sādhanā* **45**(1) (2020) 1–23.
15. L. Li, Y. Fan, M. Tse and K. Y. Lin, A review of applications in federated learning, *Comput. Ind. Eng.* **149** (2020) 106854.
16. B. Pekala, J. Szkola, K. Dyczkowski and A. Wilbik, Federated similarity-based learning with incomplete data, in *2023 IEEE Int. Conf. Fuzzy Systems (FUZZ)* (IEEE, 2023), pp. 1–6.
17. M. Przybyła-Kasperek and K. F. Marfo, Neural network used for the fusion of predictions obtained by the K-nearest neighbors algorithm based on independent data sources, *Entropy* **23**(12) (2021) 1568.
18. K. F. Marfo and M. Przybyła-Kasperek, Radial basis function network for aggregating predictions of k-nearest neighbors local models generated based on independent data sets, *Procedia Comput. Sci.* **207** (2022) 3234–3243.
19. K. F. Marfo and M. Przybyła-Kasperek, Radial basis function neural network with a centers training stage for prediction based on dispersed image data, in *Int. Conf. Computational Science* (Springer Nature Switzerland, Cham, 2023), pp. 89–103.
20. M. Przybyła-Kasperek and A. Wakulicz-Deja, A dispersed decision-making system — The use of negotiations during the dynamic generation of a system's structure, *Inf. Sci.* **288** (2014) 194–219.
21. J. Brian, Crowdsourced Mapping, UCI Machine Learning Repository (2016), https://doi.org/10.24432/C56315.
22. J. Colonna, E. Nakamura, M. Cristo and M. Gordo, Anuran Calls (MFCCs), UCI Machine Learning Repository (2017), https://doi.org/10.24432/C5CC9H.

23. D. Dua and C. Graff, UCI Machine Learning Repository, School of Information and Computer Science, University of California, Irvine, CA (2019), http://archive.ics.uci.edu/ml.
24. M. Koklu and I. A. Ozkan, Multiclass classification of dry beans using computer vision and machine learning techniques, *Comput. Electron. Agric.* **174** (2020) 105507.
25. J. P. Siebert, Vehicle recognition using rule based methods, Turing Institute Research Memorandum TIRM-87-0.18 (1987).
26. N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, SMOTE: Synthetic minority over-sampling technique, *J. Artif. Intell. Res.* **16** (2002) 321–357.
27. M. Fornasier and D. Toniolo, Fast, robust and efficient 2D pattern recognition for reassembling fragmented images, *Pattern Recognit.* **38**(11) (2005) 2074–2087.
28. Y. E. Shelepin, V. N. Chikhman and N. Foreman, Analysis of the studies of the perception of fragmented images: Global description and perception using local features, *Neurosci. Behav. Physiol.* **39**(6) (2009) 569–580.

Publication [P11]

Marfo K.F., Przybyła-Kasperek M., Decentralized Neural Network Modeling from Heterogeneous Data Sources: A Feature Mapping Approach *International Conference Information Systems Development*, 2025. doi: 10.62036/ISD.2025
URL: http://dx.doi.org/10.62036/ISD.2025.
DOI: 10.62036/ISD.2025
$\text{MEiN}_{2025} = 140$
Number of citations:

- according to Web of Science: 0
- according to Google Scholar: 0

Contributor	Description of main tasks
Kwabena Frimpong Marfo	- investigation - conceptualization and methodology - result analysis - implementation of proposed algorithm - manuscript: review and editing - visualization: creating all figures in manuscript
Małgorzata Przybyła-Kasperek	- conceptualization and methodology - result analysis - manuscript: original draft preparation - manuscript: review and editing - visualization: creating all figures in manuscript

Decentralized Neural Network Modeling from Heterogeneous Data Sources: A Feature Mapping Approach

Kwabena Frimpong Marfo
University of Silesia in Katowice
Katowice, Poland *kwabena.marfo@us.edu.pl*

Małgorzata Przybyła-Kasperek
University of Silesia in Katowice
Katowice, Poland
Constantine the Philosopher University in Nitra
Nitra Slovakia *malgorzata.przybyla-kasperek@us.edu.pl*

Abstract

This paper presents a privacy-preserving framework for distributed neural network modeling across heterogeneous data sources, where local datasets differ in both objects and attributes. To enable collaborative learning without sharing raw data or model parameters, each local decision table is independently transformed into a unified feature space using multiple dimensionality reduction techniques – Principal Component Analysis (PCA), Singular Value Decomposition (SVD), and Uniform Manifold Approximation and Projection (UMAP). Various types of neural networks – Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), Simple Recurrent Network (SIMPLE), Multilayer Perceptron (MLP) and the Radial Basis Function Network (RBF) – are trained locally, and their outputs are aggregated using soft voting (simple average) to generate final predictions. Experimental results on benchmark datasets confirm the approach's effectiveness, scalability, and robustness in decentralized learning settings.

Keywords: decentralized learning, heterogeneous data, feature transformation, neural networks, dispersed data.

1. Introduction

Modern machine learning increasingly operates in distributed environments, such as hospitals banks or research institutions, where data is siloed across different locations due to privacy, security or regulatory constraints. A growing challenge in such settings is the need to learn predictive models from fully heterogeneous data, where each local repository stores its own dataset with unique features and distinct records. These locally stored datasets – hereafter referred to as local tables – often differ not only in their content but also in structure, rendering traditional distributed or federated learning approaches inadequate.

Federated learning addresses some aspects of decentralized data by enabling collaborative model training without sharing raw data. However, it generally assumes a shared feature space across all local datasets and requires parameter synchronization during training. These assumptions break down in scenarios where local tables are structurally different, such as hospitals recording patient data with differing diagnostic tools, or environmental sensors deployed in diverse geographic regions. In such real-world cases, a more flexible and privacy-preserving approach is required.

This paper introduces a novel framework for decentralized neural network modeling over structurally heterogeneous data sources. Each local table is independently transformed into a compatible feature space using dimensionality reduction techniques, enabling interoperability without exposing original data structures. Unlike conventional federated learning, our method avoids

parameter exchange and ensures that the architecture and internal workings of local models remain opaque, reducing vulnerability to reverse-engineering or inference attacks.
The framework is general and supports a variety of transformation techniques. In this study, we use PCA, SVD and UMAP to project each local table into a k-dimensional feature space. Independent neural network models are trained locally using different architectures – MLP, SIMPLE, GRU, LSTM, and RBF networks. Their predictions are then aggregated via soft voting (simple average), avoiding raw data exchange and model synchronization.
We evaluate the proposed method on benchmark datasets from the UCI Machine Learning Repository, which were artificially partitioned into non-overlapping, heterogeneous local tables. Results demonstrate the scalability, robustness, and privacy-preserving characteristics of our approach, making it suitable for multi-institutional collaboration in sensitive domains.
The main contributions of this work are as follows:

1. A novel framework for decentralized learning over fully heterogeneous data, addressing structural disparities that traditional federated learning cannot handle.

2. Use of dimensionality reduction (PCA, SVD, UMAP) to project heterogeneous decision tables into a shared feature space while preserving privacy.

3. An architecture-agnostic training approach where each center independently trains neural network models, preserving local data confidentiality without sharing model parameters.

The rest of the paper is structured as follows: Section 2 reviews related work. Section 3 outlines the proposed framework, including feature transformation, model training and aggregation strategy. Section 4 presents experimental validation using datasets from the UC Irvine (UCI) Machine Learning Repository. Finally, Section 5 concludes with a discussion of applications, limitations and directions for future research.

2. Literature review

Distributed machine learning has advanced rapidly in recent years, with significant focus on handling data heterogeneity, ensuring privacy preservation and improving computational efficiency. Federated learning (FL), established a framework for training global models without centralizing data, thereby preserving user privacy [14]. However, standard FL approaches often assume a shared feature space across clients and struggle with non-independent and identically distributed (non-IID) data [14], [20]. Several works have proposed solutions to address these challenges, including personalized FL [23] and communication-efficient protocols [9].
Privacy-preserving techniques have been further developed to mitigate risks such as attribute inference or model inversion attacks. Differential privacy [1] and secure multi-party computation [21] have been integrated into FL frameworks to strengthen data confidentiality. Nonetheless, these methods generally assume consistent feature sets and require complex cryptographic operations, limiting scalability in heterogeneous environments [4].
Feature extraction and dimensionality reduction techniques play a pivotal role in addressing data heterogeneity. PCA, SVD, and nonlinear methods such as UMAP have been widely employed to reduce feature dimensionality while preserving relevant information [3]. Recent studies have explored distributed implementations of such techniques to enable privacy-preserving feature extraction, such as the work by Fontenla-Romero et al. [7] which uses local SVD computations in decentralized anomaly detection scenarios.
Ensemble learning and voting mechanisms have been extensively used to improve robustness and accuracy in distributed systems. Techniques such as soft voting, bagging, and boosting aggregate predictions from multiple local models to form a consensus decision without requiring parameter sharing [18], [25]. This paradigm not only enhances predictive performance but also contributes to privacy preservation by avoiding the need to exchange raw data or model

parameters [5]. Despite these advances, a critical gap remains in effectively handling fully heterogeneous datasets where neither the object sets nor the attribute spaces overlap across local data centers. Existing FL methods like FedAvg [14] assume partial alignment of data or require costly synchronization that may leak sensitive information [15]. Moreover, most feature extraction approaches presuppose consistent features across clients, limiting their applicability in realistic multi-institutional collaborations where data heterogeneity is intrinsic.
The proposed framework addresses these challenges by unifying structurally incompatible local datasets through dimensionality reduction (PCA, SVD, UMAP) and integrating independent neural network models using soft voting (simple average). While the main objective and contributions are detailed in the introduction, this work uniquely addresses the limitations of existing FL systems by combining transformation-based compatibility with model-level aggregation in a privacy-preserving, structure-independent setting.

3. Methods and models

Let $D_i = (U_i, A_i, d)$, for $i = 1, \ldots, n$, represent a set of heterogeneous decision tables from distributed data centers, where U_i is a set of objects, A_i is a set of attributes, and d is the decision attribute. These tables may differ in both objects and features – a situation common in domains like healthcare, where a patient may visit multiple hospitals with distinct but partially overlapping diagnostic attributes.
Each local dataset is transformed into a uniform feature space using multiple feature extraction techniques. Given a fixed dimensionality k, each transformation (PCA, SVD, UMAP) maps D_i into a set $\mathcal{H}_i^k = \{(U_i, \mathcal{A}_i^k, d)_1, \ldots, (U_i, \mathcal{A}_i^k, d)_T\}$. These are then horizontally concatenated to form a unified table $\mathcal{C}_i^k$, while maintaining local independence and privacy – no raw data or transformation parameters are exchanged between sites.
PCA and SVD [2], [13] provide linear projections to reduce noise and highlight global structure, while UMAP [10] captures nonlinear patterns by preserving local and global topology. Their combined use enhances representational diversity, making the transformed features robust and expressive. Each $\mathcal{C}_i^k$ is used to train an independent neural network. We evaluate five architectures: MLP, RBF, GRU, LSTM, and Simple RNN. MLPs serve as baselines for tabular data [12]; RBFs provide local sensitivity [22]; GRUs and LSTMs capture temporal patterns, with GRUs being more computationally efficient whiles Simple RNNs offer a lightweight benchmark [16]
At inference, a test sample x is transformed using the same feature extractors, yielding $\mathcal{X}^k = [x_{PCA}^k, x_{SVD}^k, x_{UMAP}^k]$ and input into each of the n local models. Their outputs are combined via soft voting (simple average) to produce the final prediction. This two-stage framework – transformation followed by local modeling and decentralized voting ensures privacy, accommodates data heterogeneity, and avoids synchronization overhead.
While transforming local tables into a uniform feature space may incur some degree of information loss, this is mitigated by using a diverse set of transformation maps (PCA, SVD, UMAP). Each method emphasizes different structural aspects of the data (linear variance, latent structure, non-linear manifold). As such, the concatenation strategy recovers complementary information, which increases the likelihood that essential patterns are preserved across maps. Figure 1 illustrates the entire pipeline.

4. Experiments

The experimental assessment of the framework is achieved by conducting tests on three separate and distinct UCI datasets, namely the Landsat Satellite – 36 attributes, 6 decision classes and 6,435 objects (4,435, training, 2,000 test) [19]; Crowdsourced Mapping – 28 attributes, 6 decision classes and 10,844 objects (7,590 training, 3,254 test) [11] and Anuran Calls – comprising 22 attributes, 4 decision classes and 7,195 objects(5,036 training, 2,159 test) [6].

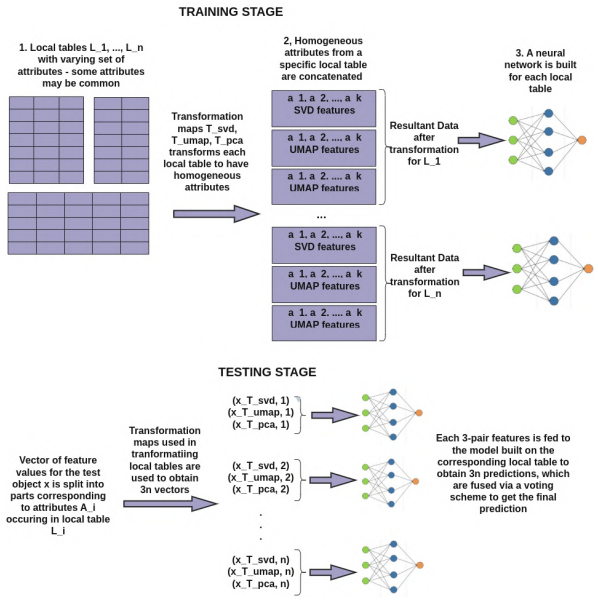


Fig. 1. Model generation and prediction for test object stages.

Each of the datasets is initially available in a non-dispersed form, with all data organized within a single decision table. The training sets are then dispersed, considering varying degrees of dispersion. Each individual data set is transformed into five different dispersed versions, with the dispersions consisting of $3, 5, 7, 9$, and 11 local tables, respectively. During the creation of these local tables, a subset of attributes from the original data set is chosen for each table. The number of attributes in each local table is significantly reduced compared to the original decision table, although certain attributes are repeated across different local tables to ensure that some overlap exists among them. This overlapping is crucial because it allows for the possibility that some local tables may share common attributes. Three transformation maps (PCA, SVD, UMAP) and five neural network models (GRU, LSTM, MLP, RBF and SIMPLE) were considered in the experiments.

Experiments varied model architectures and hyperparameters, including hidden layer depths (1-3) and the number of principal components k (ranging from 1 to the feature count of the smallest local table). Hidden layer sizes were scaled relative to the input dimension (I) – for 1 layer – $\{4, 6, 9, 12, 20\} \times I$; 2 layers – $\{(4,2), (5,3), (6,3), (8,4), (10,5), (12,6), (15,9), (20,10)\} \times I$; 3 layers – $\{(4,3,2), (6,5,4), (7,5,3), (8,4,2), (10,7,3), (12,6,3), (15,9,4), (20,10,5)\} \times I$.

Tested models included RBF (1 layer), GRU, LSTM, SIMPLE (1-2 layers), and MLP (1-3 layers). Architectural choices reflect each model's structural limitations and capabilities: recurrent models were restricted to shallow configurations to mitigate overfitting and gradient instability, while MLPs were explored more extensively due to their capacity for deep feature learning. These settings were selected to balance expressiveness and stability across architectures.

Model performance is evaluated using standard metrics: accuracy, precision, recall, and F1-score, with the latter providing a balanced measure under class imbalance. Each experiment was repeated three times, and average results are reported. Full results for MLP are provided in Table 1, while Tables 2–5 present selected outcomes for other architectures. Reported values correspond to optimal network configurations for each k, with best accuracy results highlighted in bold.

As can be seen, MLP and SIMPLE networks consistently achieved the highest accuracy and F-measure scores, particularly on the Satellite and Anuran datasets. For instance, MLP reached 0.855 accuracy on Satellite and 0.914 on Anuran, while SIMPLE achieved similar or better

Table 1. Results of prec., rec., F-m, acc, for MLP; config denotes neurons in the hidden layers.

Data set	No. tables	k	Performance metrics				config	k	Performance metrics				config
			prec.	*rec.*	F-m	*acc.*			*prec.*	*rec.*	F-m	*acc.*	
Satellite	3	1	0.515	0.51	0.477	0.51	(15;9)×I	6	0.839	0.832	0.825	0.832	(20;10;5)×I
		2	0.709	0.734	0.694	0.734	(10;7;3)×I	7	0.851	0.843	0.836	0.843	(15;9;4)×I
		3	0.822	0.825	0.811	0.825	(2;10;5)×I	8	0.857	0.853	0.847	0.853	(20;10;5)×I
		4	0.837	0.832	0.821	0.832	(20;10;5)×I	9	0.857	0.855	0.847	**0.855**	(20;10;5)×I
		5	0.833	0.828	0.817	0.828	(20;10;5)×I	10	0.856	0.855	0.847	**0.855**	(15;9;4)×I
	5	1	0.546	0.539	0.489	0.539	(20;10)×I	5	0.853	0.852	0.843	**0.852**	(15;9;4)×I
		2	0.677	0.708	0.655	0.708	(15;9;4)×I	6	0.835	0.835	0.825	0.835	(15;9;4)×I
		3	0.826	0.818	0.801	0.818	(15;9)×I	7	0.849	0.852	0.843	**0.852**	(15;9;4)×I
		4	0.84	0.836	0.821	0.836	(20;10)×I	8	0.839	0.836	0.817	0.836	20×I
	7	1	0.53	0.517	0.442	0.517	(15;9)×I	4	0.82	0.818	0.788	0.818	(15;9;4)×I
		2	0.711	0.77	0.719	0.77	(20;10;5)×I	5	0.841	0.836	0.816	**0.836**	(20;10;5)×I
		3	0.825	0.808	0.768	0.808	(20;10;5)×I						
	9	1	0.54	0.542	0.476	0.542	(20;10)×I	3	0.803	0.798	0.756	**0.798**	(20;10;5)×I
		2	0.71	0.768	0.717	0.768	(20;10;5)×I						
	11	1	0.411	0.474	0.404	0.474	(15;9)×I	3	0.821	0.813	0.773	**0.813**	(20;10;5)×I
		2	0.71	0.767	0.715	0.767	(20;10;5)×I						
Crowd sourced	3	1	0.479	0.692	0.567	0.692	4×I	6	0.782	0.778	0.711	0.778	(20;10;5)×I
		2	0.479	0.692	0.567	0.692	4×I	7	0.793	0.79	0.736	0.79	(20;10;5)×I
		3	0.588	0.732	0.635	0.732	(20;10;5)×I	8	0.756	0.784	0.725	0.784	(15;9;4)×I
		4	0.734	0.761	0.675	0.761	20×I	9	0.813	0.8	0.753	0.8	(15;9;4)×I
		5	0.712	0.768	0.692	0.768	(15;9;4)×I	10	0.8	0.802	0.757	**0.802**	(20;10;5)×I
	5	1	0.479	0.692	0.567	0.692	4×I	4	0.686	0.712	0.605	0.712	(20;10;5)×I
		2	0.479	0.692	0.567	0.692	4×I	5	0.699	0.707	0.6	0.707	(15;9;4)×I
		3	0.566	0.7	0.584	0.7	(15;9;4)×I	6	0.73	0.736	0.656	**0.736**	(20;10;5)×I
	7	1	0.479	0.692	0.567	0.692	4×I	4	0.572	0.704	0.591	0.704	(20;10;5)×I
		2	0.479	0.692	0.567	0.692	4×I	5	0.739	0.757	0.677	**0.757**	(20;10;5)×I
		3	0.567	0.699	0.582	0.699	(15;9)×I						
	9	1	0.479	0.692	0.567	**0.692**	4×I	3	0.479	0.692	0.567	**0.692**	4×I
		2	0.479	0.692	0.567	**0.692**	4×I						
	11	1	0.479	0.692	0.567	0.692	4×I	3	0.573	0.693	0.568	**0.693**	(20;10;5)×I
		2	0.479	0.692	0.567	0.692	4×I						
Anuran	3	1	0.684	0.65	0.542	0.65	(15;9)×I	5	0.896	0.904	0.898	0.904	20×I
		2	0.758	0.817	0.777	0.817	(6;3)×I	6	0.893	0.895	0.883	0.895	(15;9;4)×I
		3	0.828	0.842	0.814	0.842	(20;10)×I	7	0.906	0.914	0.908	**0.914**	12×I
		4	0.867	0.871	0.855	0.871	(15;9;4)×I	8	0.897	0.902	0.891	0.902	(12;6)×I
	5	1	0.677	0.687	0.614	0.687	(20;10)×I	4	0.847	0.861	0.847	0.861	(20;10;5)×I
		2	0.703	0.752	0.704	0.752	(20;10;5)×I	5	0.868	0.875	0.864	**0.875**	(20;10;5)×I
		3	0.798	0.823	0.787	0.823	(12;6)×I						
	7	1	0.68	0.619	0.479	0.619	(20;10;5)×I	3	0.755	0.79	0.747	0.79	20×I
		2	0.755	0.792	0.749	**0.792**	(20;10;5)×I						
	9	1	0.377	0.614	0.467	0.614	4×I	3	0.725	0.742	0.687	0.742	(20;10)×I
		2	0.742	0.771	0.725	**0.771**	(20;10;5)×I						
	11	1	0.377	0.614	0.467	0.614	4×I	2	0.742	0.764	0.715	**0.764**	(8;4;2)×I

Table 2. Results of prec., rec., F-m, acc, for LSTM; config denotes neurons in the hidden layers.

Data set	No. tables	k	Performance metrics				config	k	Performance metrics				config
			prec.	*rec.*	F-m	*acc.*			*prec.*	*rec.*	F-m	*acc.*	
Satellite	3	4	0.813	0.811	0.794	**0.811**	(20;10)×I	9	0.807	0.798	0.773	0.798	20×I
	5	1	0.545	0.541	0.467	0.541	20×I	5	0.818	0.82	0.799	**0.82**	20×I
	7	1	0.548	0.55	0.48	0.55	(20;10)×I	4	0.721	0.796	0.75	**0.796**	(20;10)×I
	9	1	0.434	0.546	0.47	0.546	20×I	3	0.713	0.762	0.718	**0.762**	20×I
	11	1	0.426	0.49	0.416	0.49	9×I	3	0.729	0.784	0.737	**0.784**	20×I
Crowd sourced	3	2	0.656	0.72	0.624	0.72	(10;5)×I	7	0.712	0.775	0.715	0.775	4×I
		5	0.669	0.768	0.684	0.768	4×I	10	0.836	0.831	0.795	**0.831**	20×I
	5	3	0.595	0.705	0.593	0.705	(10;5)×I	6	0.632	0.748	0.659	**0.748**	4×I
	7	2	0.479	0.692	0.567	0.692	4×I	5	0.634	0.736	0.652	**0.736**	(8;4)×I
	9	1	0.479	0.692	0.567	0.692	4×I	3	0.519	0.695	0.573	**0.695**	(4;2)×I
	11	1	0.479	0.692	0.567	0.692	4×I	3	0.515	0.693	0.568	**0.693**	(5;3)×I
Anuran	3	1	0.708	0.696	0.619	0.696	(18;4)×I	5	0.833	0.839	0.815	**0.839**	9×I
	7	1	0.511	0.659	0.557	0.659	(12;6)×I	3	0.757	0.813	0.776	**0.813**	(12;6)×I
	9	1	0.525	0.636	0.513	0.636	(15;9)×I	3	0.744	0.807	0.767	**0.807**	(4;2)×I
	11	1	0.479	0.692	0.567	0.692	(10;5)×I	2	0.515	0.693	0.568	**0.693**	(12;6)×I

Table 3. Results of prec., rec., F-m, acc, for GRU; config denotes neurons in the hidden layers.

Data set	No. tables	k	Performance metrics				config	k	Performance metrics				config
			prec.	*rec.*	F-m	*acc.*			*prec.*	*rec.*	F-m	*acc.*	
Satellite	3	5	0.757	0.787	0.757	0.787	20×I	10	0.86	0.855	0.843	**0.855**	20×I
	5	3	0.806	0.803	0.785	0.803	20×I	7	0.842	0.837	0.82	**0.837**	20×I
	7	2	0.683	0.723	0.676	0.723	(20;10)×I	5	0.82	0.806	0.778	**0.806**	20×I
	9	1	0.572	0.559	0.486	0.559	(15;9)×I	3	0.754	0.776	0.731	**0.776**	20×I
	11	1	0.427	0.486	0.413	0.486	20×I	3	0.731	0.79	0.741	**0.79**	(20;10)×I
Crowd sourced	3	3	0.604	0.741	0.645	0.741	(10;5)×I	8	0.819	0.817	0.781	**0.817**	20×I
		4	0.732	0.783	0.709	0.783	6×I	9	0.817	0.817	0.777	**0.817**	20×I
	5	2	0.511	0.702	0.589	0.702	(8;4)×I	5	0.727	0.774	0.703	**0.774**	4×I
	7	2	0.479	0.692	0.567	0.692	4×I	5	0.741	0.766	0.685	**0.766**	(10;5)×I
	9	1	0.479	0.692	0.567	0.692	4×I	3	0.525	0.706	0.593	**0.706**	(8;4)×I
	11	1	0.479	0.692	0.567	0.692	4×I	3	0.523	0.694	0.57	**0.694**	(10;5)×I
Anuran	3	3	0.853	0.857	0.841	0.857	9×I	7	0.884	0.884	0.876	**0.884**	12×I
	5	2	0.685	0.745	0.69	0.745	(5;3)×I	5	0.855	0.846	0.838	**0.846**	(15;9)×I
	7	2	0.764	0.83	0.791	**0.83**	(10;5)×I						
	9	1	0.666	0.695	0.633	0.695	(15;9)×I	3	0.762	0.825	0.786	**0.825**	(12;6) ×I
	11	1	0.677	0.692	0.599	0.692	(10;10)×I	3	0.771	0.826	0.794	**0.826**	(20;10)×I

performance in some configurations. GRU also performed strongly, whereas LSTM generally trailed slightly in both accuracy and computational efficiency. RBF networks showed solid results, especially at higher dimensionalities. Statistical analysis were conducted using the F-measure to compare five neural network types across 77 conditions (dataset, dispersion version, and k values). Due to the non-normal distribution of ratio-scaled data, the Friedman test was applied, revealing a significant difference among the models ($\chi^2(4, 76) = 42.70, p = 0.00001$) with MLP, SIMPLE, and GRU ranking highest as shown in Figure 2 The dimensionality pa-

Table 4. Results of prec., rec., F-m, acc, for SIMPLE; config denotes neurons in the hidden layers.

Data set	No. tables	k	Performance metrics				config	k	Performance metrics				config
			prec.	*rec.*	F-m	*acc.*			*prec.*	*rec.*	F-m	*acc.*	
Satellite	3	3	0.836	0.829	0.812	0.829	(20;10)×I	8	0.868	0.866	0.857	**0.866**	(20;10)×I
	5	1	0.511	0.542	0.467	0.542	20×I	5	0.861	0.861	0.855	**0.861**	20×I
	7	2	0.71	0.768	0.719	0.768	20×I	5	0.86	0.851	0.837	**0.851**	20×I
	9	1	0.428	0.523	0.457	0.523	(12;6)×I	3	0.806	0.797	0.757	**0.797**	(20;10)×I
	11	1	0.329	0.498	0.395	0.498	(10;5)×I	3	0.833	0.824	0.789	**0.824**	(20;10)×I
Crowd sourced	3	2	0.479	0.692	0.567	0.692	4×I	7	0.816	0.806	0.76	0.806	12×I
		3	0.712	0.743	0.652	0.743	12×I	8	0.826	0.836	0.797	**0.836**	(6;3)×I
	5	3	0.519	0.693	0.568	0.693	12×I	6	0.765	0.778	0.724	**0.778**	(20;10)×I
	7	2	0.479	0.692	0.567	0.692	4×I	5	0.761	0.771	0.709	**0.771**	20×I
	9	1	0.479	0.692	0.567	**0.692**	4×I	3	0.479	0.692	0.567	**0.692**	4×I
	11	1	0.479	0.692	0.567	0.692	4×I	3	0.565	0.697	0.577	**0.697**	(15;9)×I
Anuran	3	3	0.828	0.833	0.823	0.833	12×I	7	0.905	0.911	0.906	**0.911**	4×I
	5	2	0.699	0.734	0.68	0.734	(12;6)×I	5	0.881	0.884	0.879	**0.884**	(8;4)×I
	7	1	0.377	0.614	0.467	0.614	4×I	3	0.764	0.816	0.776	**0.816**	12×I
	9	1	0.377	0.614	0.467	0.614	4×I	3	0.84	0.814	0.776	**0.814**	20 ×I
	11	1	0.679	0.617	0.474	0.617	(15;9)×I	3	0.768	0.817	0.778	**0.817**	12×I

Table 5. Results of prec., rec., F-m, acc, for RBF; config denotes neurons in the hidden layers.

Data set	No. tables	k	Performance metrics				config	k	Performance metrics				config
			prec.	*rec.*	F-m	*acc.*			*prec.*	*rec.*	F-m	*acc.*	
Satellite	3	5	0.858	0.853	0.844	**0.853**	20×I	10	0.853	0.84	0.828	0.84	6×I
	5	2	0.701	0.731	0.681	0.731	20×I	6	0.871	0.864	0.853	**0.864**	20×I
	7	1	0.529	0.503	0.425	0.503	20×I	4	0.856	0.838	0.812	**0.838**	20×I
	9	1	0.564	0.562	0.499	0.562	20×I	3	0.837	0.817	0.779	**0.817**	20×I
	11	1	0.315	0.473	0.375	0.473	20×I	3	0.81	0.815	0.776	**0.815**	20×I
Crowd sourced	3	4	0.805	0.786	0.728	0.786	20×I	9	0.778	0.778	0.716	0.778	6×I
		5	0.772	0.783	0.724	0.783	20×I	10	0.785	0.793	0.74	**0.793**	6×I
	5	3	0.642	0.693	0.568	0.693	12×I	6	0.75	0.758	0.69	**0.758**	20×I
	7	2	0.479	0.692	0.567	0.692	4×I	5	0.782	0.761	0.689	**0.761**	20×I
	9	1	0.479	0.692	0.567	**0.692**	4×I	3	0.479	0.692	0.567	**0.692**	4×I
	11	1	0.479	0.692	0.567	0.692	4×I	3	0.711	0.695	0.571	**0.695**	20×I
Anuran	3	1	0.377	0.614	0.467	0.614	4×I	5	0.849	0.85	0.824	**0.85**	9×I
	5	2	0.644	0.696	0.642	0.696	20×I	5	0.781	0.787	0.749	**0.787**	12×I
	7	2	0.747	0.8	0.758	**0.8**	20×I						
	9	2	0.756	0.802	0.76	**0.802**	20×I						
	11	1	0.661	0.621	0.483	0.621	20×I	3	0.764	0.809	0.769	**0.809**	20×I

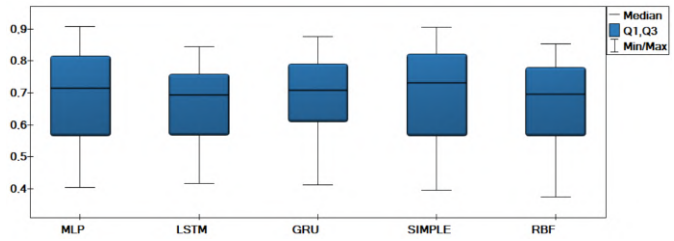


Fig. 2. Comparison of F-measure obtained for all analyzed types of neural network.

rameter k significantly affects model performance. Lower values (e.g., $k = 1, 2$) yield poor results, while performance improves consistently with higher k. This confirms that a higher-dimensional representation captures more discriminative features. To validate this, we applied the Kruskal-Wallis test to F-measure results grouped by k. Group sizes varied from 70 observations for $k \in \{1, 2, 3\}$ to 10 for $k = 9$ and 10. The test confirmed a significant effect of k on performance ($H(5) = 45.33$, $p < 0.00001$). As shown in Figure 3, higher k values correspond to increased median F-measure, indicating that richer, higher-dimensional representations enable better model performance.

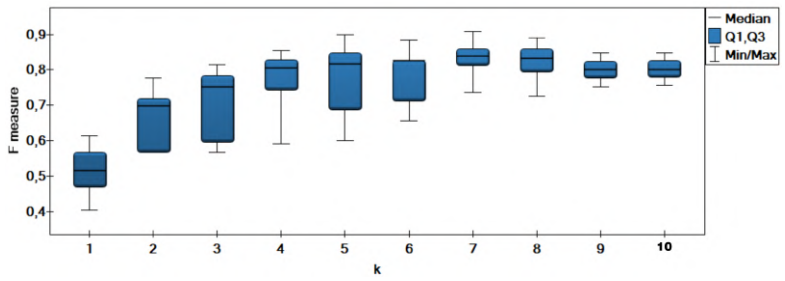


Fig. 3. Comparison of F-measure obtained for all different k values and MLP network.

Data dispersion – the number of local tables was also examined, with results showing the framework's robustness to varying levels of information separation. Performance remained stable across low and high dispersion settings, underscoring its suitability for distributed environments. Comparative analysis with recent methods [17], [24] showed comparable performance. Unlike approaches such as FedFA, which require alignment through shared features or supervised anchors [24], or federated PCA/SVD methods that transmit transformation subspaces [8], our framework avoids global parameter or feature sharing, enhancing privacy and supporting full heterogeneity.

5. Conclusion

This paper introduced a novel framework for decentralized neural network modeling across heterogeneous data sources, addressing disparities in object sets and feature spaces without sharing raw data. By mapping local datasets into equal-dimensional spaces via PCA, SVD, and UMAP, and aggregating independently trained models through soft voting (simple average), the approach maintains both structural and privacy constraints. Experiments on three UCI datasets validate the framework's effectiveness, with strong classification results from MLP, SIMPLE, and GRU models at higher dimensionalities.
However, the results are influenced by specific dataset characteristics, model architectures, and transformation choices, which may limit generalizability. Potential limitations include information loss from dimensionality reduction and computational overhead from parallel training. Future work will explore alternative transformation methods, unified table representations, and stacking-based global model integration.

References

[1] Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. p. 308–318. CCS '16, New York, NY, USA (2016)

[2] Baker, K.: Singular value decomposition tutorial. The Ohio State University 24, pp. 22 (2005)

[3] de-la Bandera, I., Palacios, D., Mendoza, J., Barco, R.: Feature extraction for dimensionality reduction in cellular networks performance analysis. Sensors 20(23) (2020)

[4] Cao, F., Liu, B., He, J., Xu, J., Xiao, Y.: Privacy preservation-based federated learning with uncertain data. Information Sciences 678, pp. 121024 (2024)

[5] Chaudhary, N., Gupta, V., Sandhir, K., Gupta, R., Chhabra, S., Singh, A.: Privacy preserving ensemble learning classification model for mental healthcare. pp. 513–518 (11 2022)

[6] Colonna, J., Nakamura, E., Cristo, M., Gordo, M.: Anuran Calls (MFCCs). UCI Machine Learning Repository (2015), DOI: https://doi.org/10.24432/C5CC9H

[7] Fontenla-Romero, O., Pérez-Sánchez, B., Guijarro-Berdiñas, B.: Dsvd-autoencoder: a scalable distributed privacy-preserving method for one-class classification. International Journal of Intelligent Systems 36(1), pp. 177–199 (2021)

[8] Hartebrodt, A., Röttger, R., Blumenthal, D.B.: Federated singular value decomposition for high-dimensional data. Data Mining and Knowledge Discovery 38(3), pp. 938–975 (2024)

[9] He, S., Zheng, J., Feng, M., Chen, Y.: Communication-efficient federated learning with adaptive consensus admm. Applied Sciences 13(9) (2023)

[10] Healy, J., McInnes, L.: Uniform manifold approximation and projection. Nature Reviews Methods Primers 4(1), pp. 82 (2024)

[11] Johnson, B.: Crowdsourced Mapping. UCI Machine Learning Repository (2016), DOI: doi.org/ 10.24432/C56315

[12] Kadra, A., Lindauer, M., Hutter, F., Grabocka, J.: Well-tuned simple nets excel on tabular datasets. arXiv preprint arXiv:2106.11189 (2021)

[13] Kurita, T.: Principal component analysis (pca). In: Computer vision: a reference guide, pp. 1013–1016. Springer (2021)

[14] McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. pp. 1273–1282. PMLR (2017)

[15] Melis, L., Song, C., De Cristofaro, E., Shmatikov, V.: Exploiting unintended feature leakage in collaborative learning. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 691–706 (2019)

[16] Mienye, I.D., Swart, T.G., Obaido, G.: Recurrent neural networks: A comprehensive review of architectures, variants, and applications. Information 15(9) (2024)

[17] Przybyła-Kasperek, M., Marfo, K.F.: A multi-layer perceptron neural network for varied conditional attributes in tabular dispersed data. PloS one 19(12), pp. e0311041 (2024)

[18] Rokach, L.: Ensemble-based classifiers. Artificial Intelligence Review 33(1), pp. 1–39 (Feb 2010)

[19] Srinivasan, A.: Statlog (Landsat Satellite). UCI Machine Learning Repository (1993), DOI: doi. org/10.24432/C55887

[20] Tan, Q., Wu, S., Tao, Y.: Privacy-enhanced federated learning for non-iid data. Mathematics 11(19) (2023)

[21] Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., Zhou, Y.: A hybrid approach to privacy-preserving federated learning. In: Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security. p. 1–11. AISec'19, New York, NY, USA (2019)

[22] Yang, Y., Wang, P., Gao, X.: A novel radial basis function neural network with high generalization performance for nonlinear process modelling. Processes 10(1), pp. 140 (2022)

[23] Zhang, G., Liu, B., Zhu, T., Ding, M., Zhou, W.: Ppfed: A privacy-preserving and personalized federated learning framework. IEEE Internet of Things Journal 11(11), pp. 19380–19393 (2024)

[24] Zhou, T., Zhang, J., Tsang, D.: Fedfa: Federated learning with feature anchors to align features and classifiers for heterogeneous data. IEEE Transactions on Mobile Computing PP, pp. 1–12 (01 2023)

[25] Zhou, Z.H.: Ensemble Methods: Foundations and Algorithms. Chapman & Hall/CRC, 1st edn. (2012)

Declarations

December 1, 2025

Małgorzata Przybyła-Kasperek

University of Silesia in Katowice

ul. Będzińska 39

41-200 Sosnowiec, Poland

Declaration

I herby declare the following contribution as an author of the following paper:

Przybyła-Kasperek M., Marfo K.F. Neural network used for the fusion of predictions obtained by the *k*-nearest neighbors algorithm based on independent data sources. *Entropy*, 23(12), 2021. doi:10.3390/e23121568

Conceptualization and methodology, result analysis, manuscript: original draft preparation, manuscript: review and editing, visualization: creating all figures in manuscript.

Małgorzata Przybyła-Kasperek

December 1, 2025

Małgorzata Przybyła-Kasperek

University of Silesia in Katowice

ul. Będzińska 39

41-200 Sosnowiec, Poland

Declaration

I herby declare the following contribution as an author of the following paper:

Przybyła-Kasperek M., Marfo K.F. Influence of noise and data characteristics on classification quality of dispersed data using neural networks on the fusion of predictions *International Conference Information Systems Development*, 2022. doi: 10.62036/ISD.2022

Conceptualization and methodology, result analysis, manuscript: original draft preparation, manuscript: review and editing, visualization: creating all figures in manuscript.

Małgorzata Przybyła-Kasperek

Małgorzata
Przybyła-Kasperek

December 1, 2025

Małgorzata Przybyła-Kasperek

University of Silesia in Katowice

ul. Będzińska 39

41-200 Sosnowiec, Poland

Declaration

I herby declare the following contribution as an author of the following paper:

Marfo K.F, Przybyła-Kasperek M. Radial basis function network for aggregating predictions of k-nearest neighbors local models generated based on independent data sets *Procedia Computer Science*, 207:3234--3243, 2022.

Conceptualization and methodology, result analysis, manuscript: original draft preparation, manuscript: review and editing, visualization: creating all figures in manuscript.

Małgorzata Przybyła-Kasperek

December 1, 2025

Małgorzata Przybyła-Kasperek

University of Silesia in Katowice

ul. Będzińska 39

41-200 Sosnowiec, Poland

Declaration

I herby declare the following contribution as an author of the following paper:

Marfo K.F., Przybyła-Kasperek M. Radial basis function neural network with a centers training stage for prediction based on dispersed image data *International Conference on Computational Science*, 10476:89-103, 2023.

Conceptualization and methodology, result analysis, manuscript: original draft preparation, manuscript: review and editing, visualization: creating all figures in manuscript.

Małgorzata Przybyła-Kasperek

Małgorzata Przybyła-Kasperek

December 1, 2025

Małgorzata Przybyła-Kasperek

University of Silesia in Katowice

ul. Będzińska 39

41-200 Sosnowiec, Poland

Declaration

I herby declare the following contribution as an author of the following paper:

Marfo K.F., Przybyła-Kasperek M. Study on the Use of Artificially Generated Objects in the Process of Training MLP Neural Networks Based on Dispersed Data *Entropy*, 25(5), 2023.

Conceptualization and methodology, result analysis, manuscript: original draft preparation, manuscript: review and editing, visualization: creating all figures in manuscript.

Małgorzata Przybyła-Kasperek

Małgorzata
Przybyła-Kasperek

December 1, 2025

Małgorzata Przybyła-Kasperek

University of Silesia in Katowice

ul. Będzińska 39

41-200 Sosnowiec, Poland

Declaration

I herby declare the following contribution as an author of the following paper:

Marfo K.F., Przybyła-Kasperek M., Sulikowski P. Fragmented Image Classification Using Local and Global Neural Networks: Investigating the Impact of the Quantity of Artificial Objects on Model Performance *International Conference on Computational Science*, 14838:280-194, 2024.

Conceptualization and methodology, result analysis, manuscript: original draft preparation, manuscript: review and editing, visualization: creating all figures in manuscript.

Małgorzata Przybyła-Kasperek

Małgorzata
Przybyła-Kasperek

December 1, 2025

Piotr Sulikowski

West Pomeranian University of Technology in Szczecin

Faculty of Computer Science and Information Technology

ul. Żołnierska 49,

71-210 Szczecin, Poland

Declaration

I hereby declare the following contribution as an author of the following paper:

Marfo K.F., Przybyła-Kasperek M., Sulikowski P. Fragmented Image Classification Using Local and Global Neural Networks: Investigating the Impact of the Quantity of Artificial Objects on Model Performance *International Conference on Computational Science*, 14838:280-194, 2024.

Result analysis, manuscript: review and editing.

Elektronicznie podpisany przez:

Piotr Stanisław Sulikowski

Data:
2025-12-1 19:2:11

Piotr Sulikowski

December 1, 2025

Małgorzata Przybyła-Kasperek

University of Silesia in Katowice

ul. Będzińska 39

41-200 Sosnowiec, Poland

Declaration

I herby declare the following contribution as an author of the following paper:

Marfo K.F., Przybyła-Kasperek M. Exploring the Impact of Object Diversity on Classification Quality in Dispersed Data Environments. *ACIIDS*, 14796:250-262, 2024.

Conceptualization and methodology, result analysis, manuscript: original draft preparation, manuscript: review and editing, visualization: creating all figures in manuscript.

Małgorzata Przybyła-Kasperek

Małgorzata
Przybyła-Kasperek

December 1, 2025

Małgorzata Przybyła-Kasperek

University of Silesia in Katowice

ul. Będzińska 39

41-200 Sosnowiec, Poland

Declaration

I herby declare the following contribution as an author of the following paper:

Marfo K.F., Przybyła-Kasperek M. Enhancing Dispersed Data Classification: A Hierarchical Model Based on Neural Networks *Proceedings of the 5th Polish Conference on Artificial Intelligence*, 154-161, 2024.

Conceptualization and methodology, result analysis, manuscript: original draft preparation, manuscript: review and editing, visualization: creating all figures in manuscript.

Małgorzata Przybyła-Kasperek

December 1, 2025

Małgorzata Przybyła-Kasperek

University of Silesia in Katowice

ul. Będzińska 39

41-200 Sosnowiec, Poland

Declaration

I herby declare the following contribution as an author of the following paper:

Przybyła-Kasperek M., Marfo K.F. A multi-layer perceptron neural network for varied conditional attributes in tabular dispersed data *PLoS ONE*, 19:1-56, 2024.

Conceptualization and methodology, result analysis, manuscript: original draft preparation, manuscript: review and editing, visualization: creating all figures in manuscript.

Małgorzata Przybyła-Kasperek

December 1, 2025

Małgorzata Przybyła-Kasperek

University of Silesia in Katowice

ul. Będzińska 39

41-200 Sosnowiec, Poland

Declaration

I herby declare the following contribution as an author of the following paper:

Marfo K.F., Przybyła-Kasperek M. Objects Diversity and its Impact on Classification Quality in Dispersed Data Environments *Vietnam Journal of Computer Science*, SN: 2196-8888 p253, 2025.

Conceptualization and methodology, result analysis, manuscript: original draft preparation, manuscript: review and editing, visualization: creating all figures in manuscript.

Małgorzata Przybyła-Kasperek

December 1, 2025

Małgorzata Przybyła-Kasperek

University of Silesia in Katowice

ul. Będzińska 39

41-200 Sosnowiec, Poland

Declaration

I herby declare the following contribution as an author of the following paper:

Marfo K.F., Przybyła-Kasperek M. Decentralized Neural Network Modeling from Heterogeneous Data Sources: A Feature Mapping Approach *International Conference Information Systems Development*, 2025. doi: 10.62036/ISD.2025

Conceptualization and methodology, result analysis, manuscript: original draft preparation, manuscript: review and editing, visualization: creating all figures in manuscript.

Małgorzata Przybyła-Kasperek

Małgorzata
Przybyła-Kasperek