Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 11, No. 1, 2023, 77-93.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

Learning a Nonlinear Combination of Generalized Heterogeneous Classifiers

Marziea Rahimi*, Amirali Taheri and Hoda Mashayekhi

Faculty of Computer Engineering, Shahrood University of Technology, Shahrood, Iran.

Article Info

Article History:
Received 09 November 2022
Revised 25 November 2022
Accepted 21 December 2022

DOI:10.22044/jadm.2022.12403.2387

Keywords:
Heterogeneous Ensemble, Classification, Neural Networks, Stacked Generalization, Classifier Fusion, Machine Learning.

**Corresponding author: marziea.rahimi@shahroodut.ac.ir(M. Rahimi).*

Abstract

Finding an effective way to combine the base learners is an essential part of constructing a heterogeneous ensemble of classifiers. In this paper, we propose a framework for constructing heterogeneous ensembles, utilizing an artificial neural network to learn a non-linear combination of the base classifiers. In the proposed framework, a set of heterogeneous classifiers work together to produce the first-level outputs in a multistep procedure where in each step, a part of the output is produced in a similar manner to cross-validation. Then these outputs are augmented using several combination functions to construct the inputs of the second-level classifier. We conduct a set of extensive experiments on 121 datasets, and compare the proposed method with other established and state-of-the-art heterogeneous methods. The results demonstrate that the proposed scheme outperforms many heterogeneous ensembles, and is superior compared to singly tuned classifiers. The proposed method is also compared with several homogeneous ensembles, and performs notably better. Our findings suggest that the improvements are even more significant on larger datasets.

1. Introduction

Classification is one of the well-known problems of machine learning. Developing accurate classifiers for real-world problems has been widely studied in the past decades. Classifiers are constructed based on a wide range of different learning algorithms. Each learning algorithm explores the hypothesis space from a different perspective. Thus by simply tuning even the best algorithm for a problem, some valuable aspects of the problem remain unexplored and ignored [1].

It has been shown that we can improve the accuracy of real-world classification problems by using ensembles rather than spending time and resources to find and tune a specific algorithm [2]. By combining classifiers and creating an ensemble learning system, several different individual perspectives are combined to make a consensus decision [3]. Following this perspective, many ensemble algorithms are introduced, such as Naïve Bayes Combiner (NBC) [4, 5], Cross-validation Accuracy Weighted Probabilistic Ensemble (CAWPE) [2], heterogeneous boosting [6], and many others [7–10]. To design an ensemble, many questions have to be answered regarding its structure, one of the most important of which is how to combine the results to make the final decision. Voting, summing [11–13], and meta-learning [14] are some of the major schemes for combining classifier decisions, among which summing and voting are used more frequently. Theoretical and practical evidence in many studies have proven the superiority of summing over voting [15]. Actually, a large number of ensemble algorithms are based on summing or in other words, weighting schemes. The central concept of many weighting schemes is to give each classifier a weight of influence based on its performance. In the meta-learning scheme, which is the least investigated among the three mentioned schemes, a second-level learning algorithm is used to learn a good combination of the base classifiers. For

homogeneous ensembles, where the base classifiers are of the same type, the meta-classifiers mainly consist of combiner trees [16, 17] and neural networks [18, 19]. In this approach, the diversity is achieved by partitioning the data.

Heterogeneous ensembles have been shown to outperform homogeneous ensembles in many applications [2, 20]. In this category, where the base classifiers use different learning algorithms to provide diversity, the meta-learning approach is represented mainly by stacked generalization (stacking). In stacking, the results of the base classifiers are combined using another classifier, often called the meta-classifier. The inputs of this second-level meta-classifier are derived from the outputs of the base classifiers. Stacking methods despite showing promising results [21], have not been thoroughly investigated and widely used due to the lack of availability of sufficient amount of data or computational resources. As these limitations are largely lifted in the recent years, the meta-learning ensemble methods are gaining more attention recently. This encouraged us to investigate the performance of stacking methods.

As mentioned earlier, the most popular and usually most successful methods for combining classifiers, both for heterogeneous and homogenous ensembles, are based on weighting schemes. This inspired us to investigate using neural networks for combining the base classifiers. Especially, by the recent successful revisit of the neural networks, which is the direct result of the availability of more enormous datasets and powerful computational platforms, it is reasonable to revisit and further investigate utilizing neural networks as the meta-classifier in ensemble systems. We propose a new stacking framework for heterogeneous ensembles, which instead of assigning constant weights to the outputs of base classifiers, uses a second-level classifier to learn the weights automatically. The second-level neural network classifier learns a mapping function from the outputs of the base classifiers to the actual target values. The proposed framework consists of three steps: generating the probabilistic outputs of the base classifiers and constructing the meta-data, building the second-level meta-classifier, and regenerating the base classifiers. Instead of simply partitioning the dataset among the base classifiers, which is the general approach in ensemble learning, we utilize the iterative folding scheme of stacked generalization to produce the outputs of the first-level classifiers. In addition, the proposed framework investigates several different combination functions to construct the meta-data as inputs of the second-level classifier. We empirically investigate the performance of the proposed stacking framework by evaluating it on 121 UCI benchmark datasets. The results are compared to different state-of-the-art, and established ensemble methods, like Majority Vote (MV), Weighted MV (WMV), several heterogeneous ensembles such as CAWPE [2], Recall Combiner (RC), [4] and Stacking with Multi-response Linear Regression (SMLR) [22], and several homogeneous ensembles like Random Forest [23] and XGBoost [24]. Through these investigations, we show that the proposed heterogeneous ensemble framework, while using a neural network as the second-level classifier, is able to outperform many other state-of-the-art and established weighting scheme and stacking methods. The contributions and highlights of our work can be summarized as follows:

- Proposing a new stacking framework for heterogeneous ensemble learning
- Combining the idea of weighting schemes and stacked generalization using a neural network
- Introducing new approaches for constructing and augmenting the meta-data as inputs of the second-level classifier, and demonstrating that some of these approaches can improve the classification accuracy
- Conducting rigorous and detailed experiments to demonstrate that the proposed framework can outperform different state of the art and benchmark ensemble models

In the rest of the paper, an overview of the related works on heterogeneous and homogenous ensembles focusing on meta-learning methods is provided in Section 2. Section 3 introduces a general layout for weighting scheme methods, and provides a brief introduction to some of the methods used for comparison in the experiments. The proposed model is introduced in Section 4. A set of comprehensive experiments are reported in Section 5, and finally, conclusion is provided in Section 6.

2. Literature Review

Ensemble learning algorithms can be categorized into two main groups of heterogeneous or homogeneous ensembles, according to the employed base classifiers. Homogenous

ensembles refer to ensemble algorithms that use the same type of base classifiers with different training data, whereas heterogonous ensembles combine the results of different base classifiers with the same training data. Recent studies suggest that a heterogeneous ensemble of a set of diverse classifiers can perform more efficiently compared to homogeneous ensembles [2, 20]. In both the heterogeneous and homogeneous categories, classifiers should be diverse. Diversity is one of the key elements that has a crucial effect on the accuracy of the ensemble [25]. In homogeneous methods, diversity is achieved either by varying the training data of the base classifiers as performed in bagging [26] and boosting [27] or by varying the feature subset that each base classifier is built on, as in ensemble feature selection methods [28]. Heterogeneous methods ensure diversity by varying the base classifiers. The difference between classifiers is either in their type of learning algorithms or in their hyper-parameters.

Our focus in this paper is on the meta-learning methods in the heterogeneous category. In the following, we will discuss the ensemble methods and some of their major examples in more detail focusing on heterogeneous methods.

The existing heterogeneous ensembles can be divided into three main categories of fusion methods, selection methods, and meta-learning, considering the algorithm employed for combining the results of the base classifiers. The former consists mostly of weighting schemes. Weighting schema algorithms are one of the first types of ensemble methods proposed and used in heterogeneous as well as homogeneous ensembles. Such methods assign weights to the base classifiers, and classify a sample using a weighted combination of base outcomes. Methods such as majority vote and weighted majority vote [29, 30] are still used in many applications of ensemble methods. Recall combiner [4] and Naïve Byes combiner [4] are other established heterogeneous examples of weighting scheme methods. CAWPE [2] is a recent weighting scheme that sums up the advantages of the weighting schemes and suggest four properties that if used alongside each other, could make a powerful algorithm that outperforms other, even more complicated, heterogeneous and homogenous methods.

Ensemble selection algorithms are focused on selecting the best possible combination of base algorithms. The idea of selection is applicable in both homogeneous [31] and heterogeneous [32–34] ensembles. Some methods [35] have also been proposed that are a combination of selection and fusion. Pick Best (PB) is the most straightforward algorithm in this category, which evaluates each of the base classifiers, typically by cross-validation, and selects the best one. As argued by Dzeroski and Zenko [36], pick best is a suitable baseline for comparison because of its simplicity and good performance. This category of models is less prone to overfitting. However, they usually are effective when applied to a large number of base classifiers, and therefore, are usually more complex, in both time and space, compared to the other categories of heterogeneous methods [37, 38].

The meta-learning scheme, which has been mentioned before, is the least investigated among the three ensemble schemes; a learning algorithm is used to learn how to effectively combine the base classifiers using their outcomes as meta-data. For homogeneous ensembles, the commonly used meta-learning algorithms are the arbiter and combiner trees [16, 17], and neural networks [18, 19]. When using neural networks in homogeneous ensembles, meta-learning sometimes follows an end-to-end fashion, which is similar in spirit to stacking in which a combiner is trained in a generalized version of cross validation. In stacking, the base models are trained and then fixed. In end-to-end ensemble methods [19], however, the combiner participates in the training of the base neural networks [39].

Meta-learning in heterogeneous ensembles has been realized in two different types; stacking and grading [40]. Grading [41] is a type of meta-learning in which predictions of each of the base classifiers are marked as correct or incorrect, and then these graded predictions are assumed to form two meta-classes. Corresponding to each base classifier, a meta-classifier is trained based on this meta-data. The aim is to gain the ability to predict when a base classifier will misclassify. For classifying each unseen instance, the predictions of base classifiers are gathered and each of them is marked as correct or incorrect. The final classification is derived from the results of those base classifiers marked as correct. Although grading methods [41, 42] are inspired by stacking, they are based on selection, contrary to stacking which follows the fusion scheme. Grading suffers from the problem of imbalanced data as in the meta-level, the number of correctly guessed instances is usually many more than the incorrect ones [43]. On the other hand, [44] shows that grading could be reduced to a special case of stacking.

Stacking algorithms use a second-level classifier to learn the best possible combination model. The second-level classifier uses cross-validation to learn from the data provided by the base classifiers. This scheme is applied on ensembles of both the same [45] and different learner types [14]. In the case of the former, the diversity is achieved by using different parameters or hyper-parameters for each of the base classifiers. Stacking, despite showing promising performance, has received little attention.

Our approach aiming to use a neural network as a combining algorithm is considered a stacking algorithm. One of the first stacking algorithms [22] uses multi-response linear regression as the meta-classifier. This method is improved in another research [36] by augmenting the class probability distributions, which are obtained from the base classifiers by additional attributes. Another example of staking algorithms is GA-stacking [46], which employs the Genetic Algorithm (GA) to select the meta-classifier. Similarly, ABC-stacking [47] exploits the ability of the Artificial Bee Colony (ABC) to select the meta-classifier at the stacking level. Non-strict stacking [48] is another version of stacking, which can be considered as a combination of same-type and different-type stacking methods. In this method, each base classifier can participate in the ensemble with many duplicates, which are constructed using different feature subsets of the original data. Xia *et al*. have proposed to combine bagging and stacking to improve diversity [49]. Some models [50] use other ensembles as the base level in the stacked generalization scheme. Stacking methods have used many different types of models as meta-classifiers. For example, [51] and [52] use logistic regression as the meta-classifier and [53], investigates tree-based methods. However, to the best of our knowledge, the use neural networks as a meta-classifier, in heterogeneous stacking ensembles, has not been thoroughly investigated.

Neural networks have been used as the meta-classier for homogeneous ensembles of neural networks [45] or recently, ensembles of deep neural networks [54]. In another example of such models [55], six different classifiers such as nearest neighbor, SVM and neural networks are investigated as a meta-classifier to combine several convolutional neural networks in a stacking scheme, and the best results were achieved by MLP.

In this paper, we propose a new heterogeneous ensemble method, which uses a feed-forward neural network to learn to combine the outputs of the base classifiers. On the other hand, stacking is usually constructed using a few numbers of base classifiers, and as the meta-level data is constructed by concatenating the outputs of these classifiers, the number of features in meta-level space is limited. Thus we have proposed and investigated several methods to enrich the meta-level feature space. We have also considered averaging as an alternative to concatenation.

3. Preliminaries

Our proposed framework is based on a stacking method, which inspired by the success of the weighting and voting approaches, combines the base-level models in a meta-learning step. Therefore, in this section, we discuss stacking and some voting schemes in more details. The methods discussed here are used as baselines for comparison in our experiments.

Table 1. List of notations used in the algorithm description.

Notation	*Description*
C , C_i	Set of base classifiers, the i^{th} classifier
A_i	The i^{th} learning algorithm
P	Number of base classifiers
K	Number of classes
x_j	A data point
$\boldsymbol{y}^i$, $\hat{\boldsymbol{y}}_i^j$	Output vectors for data point x^j assigned by the ensemble, base classifier C_i
l^j , l_i^j	Label assigned to x^j by the ensemble, C_i
M	Number of folds in the first-level learning
$\tilde{\boldsymbol{x}}_j$	The meta-data vector of x_j
D^{meta}	The set of all meta-data vectors

3.1. Weighting scheme

As suggested and justified by [4], we will use the following voting schemes to compare our model with: Majority Vote (MV), Weighted Majority Vote (WMV), Recall Combiner (RC), and Naïve Bayes Combiner (NBC). We will also compare the proposed model with the recently suggested weighting approach CAWPE [2].

The following terminology is used to describe the methods. Let us assume that there is an ensemble of P base classifiers $C=\{C_1,C_2,\ldots,C_P\}$ to classify data into K classes $1\ldots K$. A weight w_i is assigned to each classifier C_i. The probabilistic output of the base classifier C_i for a data point x_j is a vector $\hat{\boldsymbol{y}}_i^j$ of size K, where the k^{th}

element $\hat{y}_{i,k}^{j}$ is the probability assigned to class k. The label proposed by this classifier for x_j is denoted as $l_i^j \in \{1,\ldots K\}$. Similarly, the probabilistic output of the ensemble for input x_j is a vector $\boldsymbol{y}^i$, where the k^{th} element y_k^j is the probability assigned to class k. The label proposed by the ensemble x_j is denoted as l^j. The list of notations is provided in Table 1.

There are two approaches for combining the outputs of the base classifiers when using a weighting scheme. In the first approach, the predicted labels are directly used, and the weighted sum of the base outputs are calculated for each label k as follows:

$$y_k^j = \sum_{i=1}^{P} w_i \delta\left(l_i^j, k\right) \quad (1)$$

where $\delta\left(l_i^j, k\right)$ is the Kronecker delta, being one if the base classifier C_i assigns label k to x^j, and zero otherwise. The second approach employs the actual probabilistic outputs of the classifiers using the following equation:

$$y_k^j = \sum_{i=1}^{P} w_i \hat{y}_{i,k}^{j} \quad (1)$$

In both scenarios, the actual predicted label assigned to x_j by the ensemble is calculated by the following equation:

$$l^j = \underset{k}{\operatorname{argmax}}\, y_k^j \quad (2)$$

In other words, the proposed label of the ensemble is obtained by selecting the most weighted class. Now we can describe the mentioned weighting schemes as follows.

In other words, the proposed label of the ensemble is obtained by selecting the most weighted class. Now we can describe the mentioned weighting schemes as follows:

- MV: Majority vote uses Equation (1) and sets $w_i = 1$ for all the base classifiers.
- WMV: Weighted MV usually uses Equation (1) with the classifier weights being computed based on their performance (accuracy) on the training data.
- RC: Recall Combiner uses Equation (1) while considering a separate $w_{i,k}$ per each classifier and class label, based on the class-specific recalls [4].
- NBC: Naïve Bayes Combiner is very similar to Equation (1), with the weights being class-dependent and calculated based on the confusion matrix for each class [5].
- CAWPE: CAWPE uses class probabilities as depicted in Equation (2), with the weights being class-dependent and the ensemble accuracy estimated through cross-validation [2].

In this paper, we aim to demonstrate that neural networks, in combination with stack generalization can be used to effectively optimize the weights and learn a non-linear combination of the base classifiers' outputs.

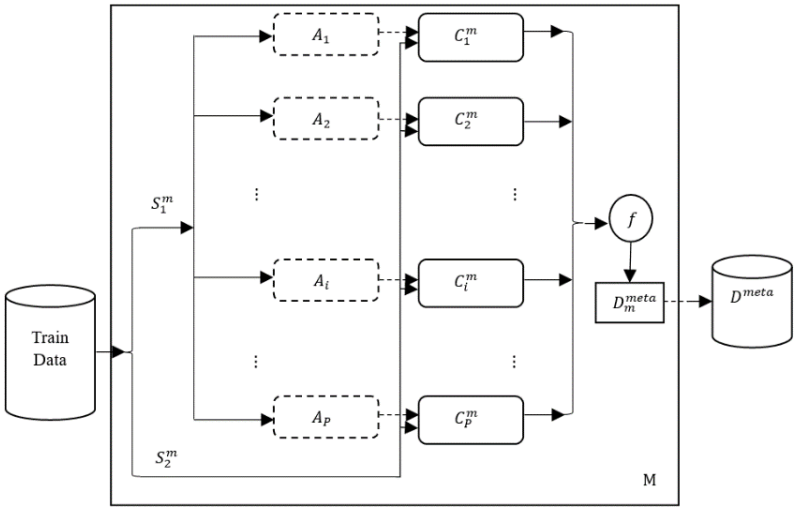
3.2. Stacking

Stacking is a heterogeneous meta-learning scheme in which the input (first-level) data are mapped into an intermediary space through a process similar to cross-validation. The meta-level classifier is trained in this new space. The first-level data is divided into M folds. Then in M iterations, the next-level data are generated. In each iteration, one of the folds is held out and the remaining $M-1$ are used to train the first-level models, i.e., the base classifiers. Then the held-out fold is presented to the trained classifiers and the output is used as one part of the next-level data. At the end of the iterations, the number of instances in the second-level data is equal to the first-level data. The target values are shared between the two levels. Any classifier can be trained using this intermediary data as the meta-classifier. In our proposed model, the weights are learned using a feed-forward neural network. The base classifiers are assumed to generate probabilistic outputs for the K classes. Otherwise, such probabilities can be calculated using the probability calibration based on the Platt's logistic model [56].

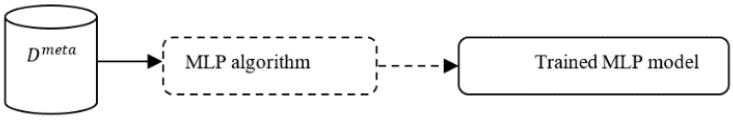
4. Proposed Heterogeneous Ensemble Framework

In this paper, we introduce a new heterogeneous stacking framework, which we refer to as Metanet. This framework uses a feed-forward neural network to learn to combine base classifiers outputs. The motivation for this proposal is the established success of the heterogeneous ensembles that use a weighting

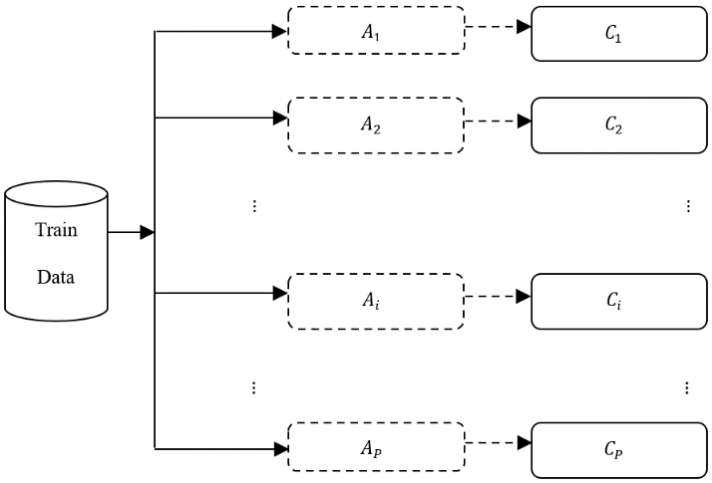
scheme to combine the outputs of the base classifiers.



(a)- First step: the training dataset is divided into M parts that lead to M iterations for generating the second-level data. In the m^{th} iteration, the m^{th} part of the second-level data is generated.



(b)- Second step: the meta-classifier is trained using the meta-data



(c)- Third step: the base classifiers are regenerated using the entire train data

Figure 1. The three steps of stack generalization.

Our model uses a multilayer neural network while satisfying the following properties suggested by Large *et al*. [2]:

- Using a small set of diverse classifiers instead of a large number of weak ones of the same learning paradigm
- The inputs of the second (combination) level are based on the probability estimations produced by the base classifiers
- The weights are obtained considering the quality of the base classifiers
- The weights are obtained also by emphasizing the difference between the base classifiers in quality

The first two conditions are exactly followed. Large *et al*. satisfy the last two conditions by simply setting weights based on the quality of the base-classifiers, and amplifying the differences by rising them to the power of a positive integer value. In our method, in contrast, we satisfy them by using a feed-forward neural network. The network combines the probability values generated by base classifiers using weights that

are learned automatically. This means that the qualities of the base classifiers are taken into account by the network through the learning process, and the magnitude of the weights are determined respectively. However, here we are not learning a simple linear combination but a non-linear mapping function that is specified by the weights of the neural network.

Algorithm 1- Training the heterogeneous model

Input: D^{train}, P learning algorithms, the combination function f

Output: The set of base classifiers C, the meta-classifier C^{meta}

1 Initialize P classifiers $C=\{C_1,C_2,\ldots,C_P\}$ based on learning algorithms A_i, $i=1\ldots P$

2 Partition D^{train} into M exclusive subsets: D_m^{train}, $m=1\ldots M$

3 For $m=1\ldots..M$

4 $S_1^m=\bigcup_{j\neq m} D_m^{train}$

5 $S_2^m=D_m^{train}$

6 Train each classifier C_i using S_1^m

7 Apply each classifier C_i on S_2^m to obtain class probabilities $\hat{\mathbf{y}}_i^j$, $i=1\ldots P, j=1\ldots\left|S_2^m\right|$

8 Apply combination function f to vectors $\hat{\mathbf{y}}_i^j$ to obtain D_m^{meta}

9 EndFor

10 $D^{meta}=\bigcup_{m=1}^{M} D_m^{meta}$

11 Train meta-classifier C^{meta} on D^{meta}

12 Regenerate the set of base classifiers C using D^{train}

13 Output C and C^{meta}

The proposed method follows the stacking scheme in which, the first-level data is divided into M folds. Then in M iterations, the second-level data is generated such that in each iteration, one of the folds is held out and the remaining $M-1$ folds are used to train the first-level models. Then the held-out fold is fed to the trained classifiers and the output is used as one of the M parts of the second-level data. After the iterations are finished, the number of instances in the second-level data equals to the number of instances in the first-level data. The target values are shared between the two levels. The input of the neural network meta-classifier, called the "meta-data" is constructed using this intermediary data. All of the instances are used to train the neural network. The meta-data itself can be divided into folds to be used in a cross-validation scheme to generalize the training process of the neural network in the second-level. After training the neural network, the entire first-level data is used to re-generate the base classifiers, which can be used to classify the unseen data. This process has three distinct steps: training the base classifiers to generate the second-level data and construct the meta-data, training the second-level classifier, and regenerating the base-level models.

The three steps, as elaborated above, are depicted in Figure 1. In the first step, the training data D^{train} is divided into M parts (folds) that leads to M iterations for generating the meta-data D^{meta}. In the m^{th} iteration, the m^{th} part of the meta-data is generated. In each iteration m, one part (S_2^m) is held out and the remaining $M-1$ parts (collectively shown as S_1^m) are used to train each base-level classifier C_i^m using the learning algorithm A_i. Then the held-out part is presented to the classifiers to generate the m^{th} part of the meta-data D_m^{meta}. Function f is used to combine the outputs of the base classifiers and generate the meta-data. In the second step the meta-data is used to train the meta-classifier C^{meta}. In the third step, the entire original train data is used to regenerate the base-classifiers, which can then be

used for the prediction step. The three steps are described in Algorithm 1.

For integrating the outputs of the first-level classifiers and constructing the meta-data, we use different combination schemes. If the initial training data is of size N, the constructed meta-data is of the same size. Let $\tilde{\boldsymbol{x}}_j$ denote the meta-data vector corresponding to the data instance x_j. For a training set of size N, the final meta-data is defined as $D^{meta} = \bigcup_{j=1}^{N} \tilde{\boldsymbol{x}}_j$. In the concatenating scheme, the class probabilities generated by the base classifiers are concatenated to generate $\tilde{\boldsymbol{x}}_j^{CP}$ as a vector of size $P \times K$:

$$\tilde{\boldsymbol{x}}_j^{cp} = \left(\hat{\boldsymbol{y}}_1^j, \hat{\boldsymbol{y}}_2^j, \ldots, \hat{\boldsymbol{y}}_P^j \right) \tag{4}$$

We have also considered integrating the outputs of the first-level classifiers using the average function as shown in **Error! Reference source not found.** The resulting meta-data vector $\tilde{\boldsymbol{x}}_j^{avg}$ is of size K.

$$\tilde{\boldsymbol{x}}_j^{avg} = \left(\underset{\mathrm{i}}{\mathrm{avg}} \left\{ \hat{y}_{i,1}^j \right\}, \underset{\mathrm{i}}{\mathrm{avg}} \left\{ \hat{y}_{i,2}^j \right\}, \ldots, \underset{\mathrm{i}}{\mathrm{avg}} \left\{ \hat{y}_{i,K}^j \right\} \right) \tag{5}$$

The performance of the model using each one of these functions is evaluated in our experiments and the results are reported in Section 5.4.

When constructing the meta-data by concatenation, the number of constructed features is limited to the number of base-classifiers times the number of classes. This may be limiting especially when the number of classes is small. Therefore, we may use the original data besides the generated class probabilities to enrich the feature space:

$$\tilde{\boldsymbol{x}}_j \leftarrow \left(\tilde{\boldsymbol{x}}_j, x_j \right) \tag{3}$$

This approach can be used with both the concatenation of class probabilities (4) and class probability averages (5). Another approach to enrich the meta-data feature space is to concatenate both the class probabilities and class probability averages for each data:

$$\tilde{\boldsymbol{x}}_j^{cp+avg} \leftarrow \left(\tilde{\boldsymbol{x}}_j^{avg}, \tilde{\boldsymbol{x}}_j^{cp} \right) \tag{7}$$

We construct four different compositions according to the mentioned functions. The first composition just concatenates the first-level outputs (class predictions) as the meta-level data. We will name this primary composition as “Class”. In another composition that we will call “Class + Base”, the concatenated first-level outputs are augmented with the input data to construct the meta-data. In another composition, called “Mean + Class” the mean vector of first-level outputs is calculated and then the concatenated first-level output is augmented by the mean vector to construct the meta-data. The last composition uses the mean vector augmented by the input as the meta-data. This composition is called “Mean + Base”. Figure 2 shows these four models. In the third step of the framework, the base classifiers are regenerated using the entire train data. This gives them a more generalized prediction power. When the labels of a dataset are to be predicted, the data is presented to all of the base-level models and the class probabilities are generated. These probabilities are used to construct the meta-data as the next level's input. The probabilities are fed to the meta-classifier and the trained neural network decides the class label based on this meta-data.

5. Experiments

In this section, we first describe the dataset and the experimental settings. Then the results obtained are presented. We have followed the same settings used by [2]. The experiments are performed on 121 datasets. All the results on each dataset are obtained by averaging over 30 different test and train splits generated using stratified sampling. In all the splits the ratio of test to train examples is kept approximately the same. We use five different algorithms as our base classifiers. The results of the suggested ensemble are compared to each of the base classifiers individually, 11 other heterogeneous ensembles, and five homogeneous ensembles.

5.1. Dataset

We use 121 datasets from the UCI archive for our experiments. The datasets cover a variety of classification problems, the smallest dataset contains ten samples, the largest one contains 130064 samples, and they have an average of 4554.504 samples. The average number of attributes per dataset is 28.842, and the average number of classes per dataset is 6.851. As mentioned earlier, for each dataset, 30 different test and train splits are generated with approximately equal sizes using stratified sampling. In other words, the test and train sets, each cover 50% of the data. We have used the exact splits, which are provided by [2].

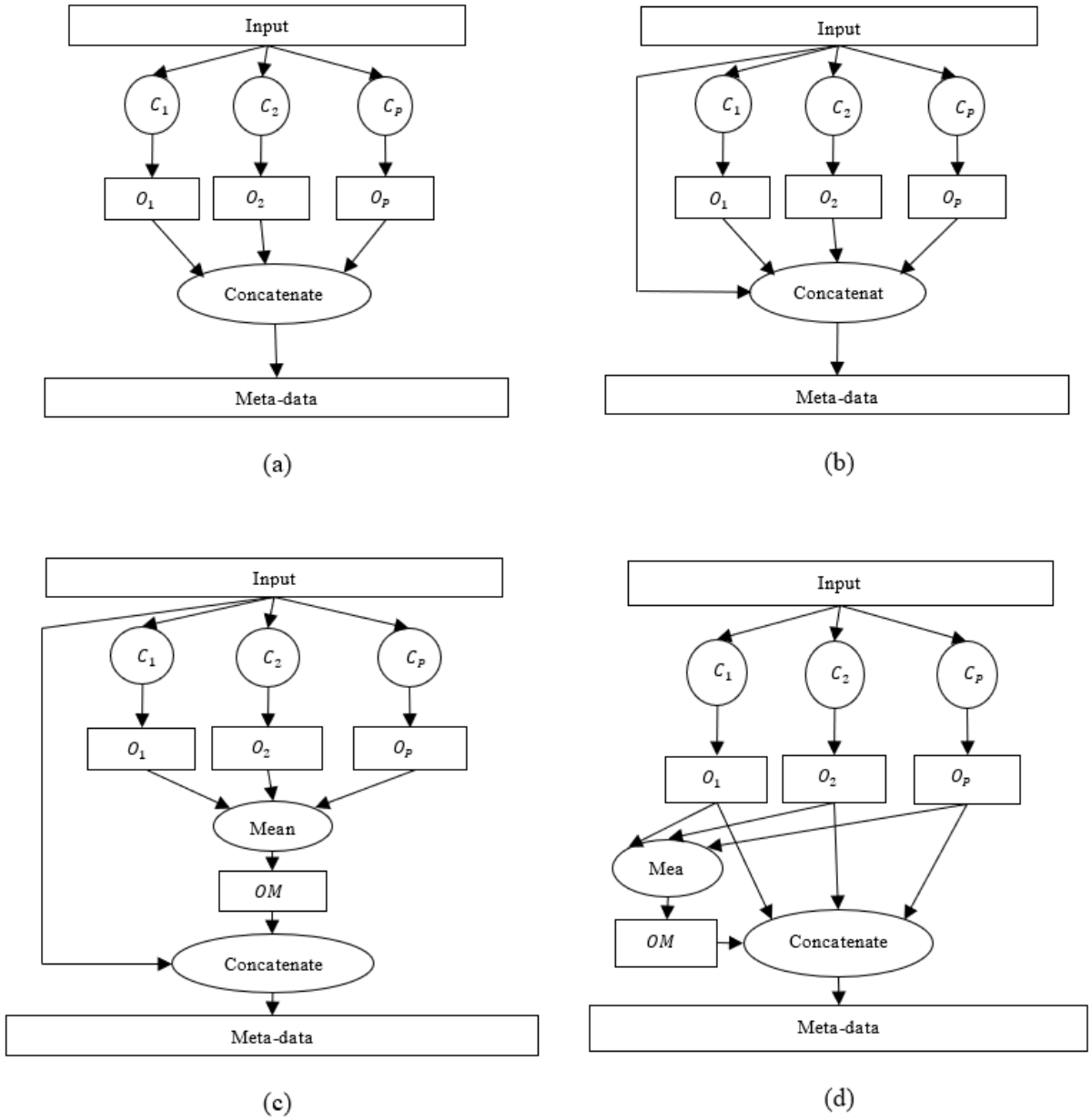


Figure 2. Four different proposed composition to construct meta-data. From (a) to (d), "Class", "Class+Base", "Mean+Base" and "Mean+Class". OM **denotes the mean of the class probabilities** O_i **.**

As the number of datasets is large, reporting the sole average values of different metrics may not provide a complete insight into algorithm performance. Therefore, we further conduct a clustering task to provide a meaningful partitioning of the datasets into several groups. The performance of the model is then evaluated individually in these groups. These results are presented in Section 5.4.5.

5.2. Evaluation Metrics

The classifiers are evaluated using four different metrics: accuracy, balanced accuracy, the area under the ROC curve (AUC), and negative log likelihood (NLL). The reported result for each classifier is averaged over 3630 experiments. We use balanced accuracy to deal with possible class imbalances when evaluating the accuracy of our method. The AUC metric is not sensitive to class imbalances; however, it is able to provide us with an evaluation of probability estimates of the models and their ability to separate the classes regardless of the selected threshold. For problems with two classes, we treated the minority class as a positive outcome. For multi-class problems, we calculate the AUC for each class and weight it by the class frequency in the train data, as recommended by [57]. NLL evaluates the likelihoods. The higher the probabilities assigned to the correct classes, the better the classifier. For the values of NLL, contrary to the others, less means better.

5.3. Experimental settings

As mentioned before, we use a set of diverse base classifiers, namely Logistic Regression (LR), C4.5 decision tree, linear Support Vector Machine (SVML), Nearest Neighbor (NN), and a Single Hidden Layer Perceptron (MLP) as suggested in many other studies [2]. As mentioned earlier, the meta-classifier is a feedforward neural network. It is a multi-layer perceptron with three layers. In other words, the network has only one hidden layer as our contribution in this paper is mainly focused on demonstrating the ability of a simple neural network to learn a heterogeneous ensemble and not finding the best network structure.

Table 2. List of 10 heterogeneous and five homogeneous ensembles used for performance comparison.

Heterogeneous ensemble methods	
Pick Best	PB
Majority Vote	MV
Weighted Majority Vote	WMV
Multi-Response Linear Regression [22]	SMLR
Naïve Bayes Combiner [5]	NBC
Ensemble Selection[58]	ES
Multi-Response Linear Regression on Extended Features [36]	SMLRE
Multi-Response Model Trees [36]	SMM5
Recall Combiner [4]	RC
Cross-validation Accuracy Weighted Probabilistic Ensemble [2]	CAWPE
Grading [59]	GRAD
Homogeneous Ensemble Methods	
Bootstrap Aggregating [60]	Bagging
Adaptive Boosting [61]	AdaBoost
Random Forests [23]	RandF
Additive Logistic Regression [62]	LogitBoost
Gradient boosting [24]	XGBoost
Rotation forest [63]	ROF

The size of the hidden layer is heuristically set as half the overall number of classes and features in the problem at hand. Hyperbolic tangent function (tanh) is used as the activation function in the hidden layer.

If unspecified in any experiment, the meta-data used for training the network is by default the concatenated class probabilities ($\tilde{\boldsymbol{x}}_j^{cp}$). The algorithm is implemented in python using Keras[1]. The implementation is publicly accessible[2].

The performance of the proposed model is compared with each one of these classifiers individually. The reported values are the mean values of the performances of each classifier on the 121 UCI datasets and on 30 different test and train splits for each dataset. The proposed model is also compared with established and state-of-the-art homogeneous and heterogeneous ensemble models. The list of these models is available in Table 2.

In the experiments, after comparing the performance of the proposed framework with other models, we investigate the effect of different combination functions in enriching the meta-data feature space.

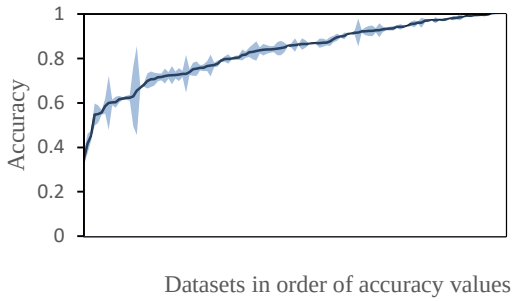


Figure 3. Average accuracy values and standard deviations of Metanet on 121 datasets.

5.4. Results

In this section, we will describe the conducted experiments and present the results of these experiments. These experiments are conducted on the mentioned 121 datasets.

As a primary evaluation, we measure the accuracy of Metanet on 121 datasets. Figure 3 shows the mean accuracy values (solid line) on the 30 splits of each dataset, and the standard deviations (SD) are shown as shadows around the accuracies.

1https://github.com/keras-team/keras
2https://github.com/amiralitaheri/tsml/blob/metanet/Metanet_Python.ipynb

The problems in this figure are sorted according to their accuracies. The largest SD values belong to the problems "Plant-shape" and "Congressional". The SD values are 0.2 and 0.135, respectively. The former problem is a two-class problem with 29 features and only 10 instances, while the latter is another two-class problem with four features and only 16 instances. These are actually the two smallest datasets among the 121 datasets. According to the small size of instances in these datasets, it is not unexpected for these two problems to be unstable when trained on different splits of the training data.

The smallest accuracy obtained by Metanet is 0.381 that belongs to the "heart-va" problem. It is a five-class problem with 12 features and 200 instances. The largest accuracy is one which belongs to "acute-inflammation". It is a two-class problem with six features and 120 instances. The results show that the accuracies obtained by the proposed ensemble framework are mostly stable on different datasets.

5.4.1. Comparison with base classifiers

The proposed model is an ensemble of five classifiers, which are called base classifiers throughout this manuscript. Naturally, the proposed model has to perform better than these base classifiers for it to be justifiable and worthy of further investigations and evaluations.

Table 3 shows the average results of applying the base classifiers on 121 UCI datasets using the setting explained in Section 5.3. One can see that metanet significantly outperforms base classifiers in terms of accuracy, balanced accuracy, the area under the ROC curve, and negative log-likelihood. This justifies the claim that a heterogeneous ensemble can perform better than multiple individual classifiers.

Table 3. Comparison between metanet and base classifiers.

Algorithm	Accuracy	Balanced accuracy	AUC	NLL
Logistic	0.762	0.691	0.841	8.134
C4.5	0.770	0.699	0.736	1.161
MLP	0.786	0.712	0.860	1.297
NN	0.784	0.697	0.798	1.116
SVML	0.771	0.694	0.849	1.073
Metanet	**0.831**	**0.756**	**0.881**	**0.725**

5.4.2. Comparison with heterogeneous ensembles

The proposed model is compared to the other heterogeneous ensembles constructed using the same set of base classifiers. The experiments are performed under the settings explained in Section 5.3, and the results are depicted in Table 4.

Table 4. Comparison between metanet and heterogeneous ensembles.

Algorithm	Accuracy	Balanced accuracy	AUC	NLL
CAWPE	0.815	0.742	**0.884**	**0.704**
ES	0.810	0.734	0.813	0.884
MV	0.805	0.727	0.808	0.877
NBC	0.807	0.740	0.820	0.999
PB	0.771	0.694	0.847	0.950
RC	0.805	0.712	0.811	0.912
SMLR	0.805	0.728	0.737	1.144
SMLRE	0.786	0.712	0.734	1.251
SMM5	0.805	0.729	0.744	1.046
WMV	0.808	0.730	0.814	0.872
GRAD	0.496	0.411	0.516	2.005
Metanet	**0.831**	**0.756**	**0.881**	0.725

As observed, metanet has the best accuracy among all heterogeneous ensembles. In terms of AUC it is comparable with CAWPE, while outperforming the other methods. However, CAWPE has a better negative log-likelihood than the proposed model while it is better than the other methods. However, according to our observations, NLL could be improved by using more information to achieve better probability estimates. This will be further discussed in the Section 5.4.4.

The results confirm that using a heterogeneous ensemble with a neural network meta-learner can achieve a notable performance which outperforms other established heterogeneous models.

As depicted above, the proposed model is more accurate on average than many other types of heterogeneous ensemble models, the most recent of which is CAWPE. For more clarification, we have investigated the performance of CAWPE and Metanet more closely. Figure 4 shows the differences in the accuracies of the Metanet and CAWPE models on each of the problems in 121 UCI datasets. As one can see in this figure, the proposed model is more accurate in most of the cases.

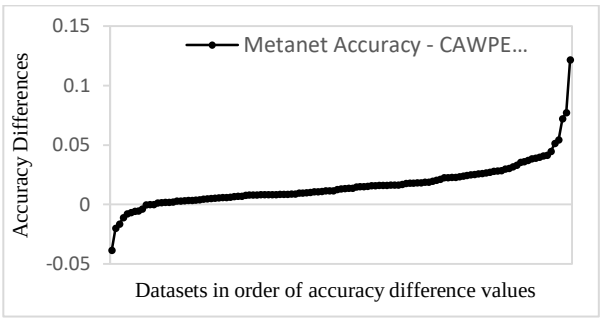


Figure 4. A difference between Metanet accuracy and CAWPE accuracy in increasing order.

Accurately speaking, the accuracy of CAWPE is better in only 10 datasets out of the total 121 datasets. The average number of instances in these 10 datasets is 587.1, meaning that these are small

datasets. The average size of the other 111 datasets where Metanet performs better is 4911.9. This can be considered as an indicating point that the proposed model has the potential to perform even better on bigger datasets.

5.4.3. Comparison with homogeneous ensembles

The performance of Metanet is compared with five different homogeneous ensembles, all of which are constructed using 500 different instances of the same base classifier. The results are reported in Table 5.

As depicted in Table 5, Metanet is also more accurate than homogeneous ensembles. In terms of AUC, the proposed model outperforms four of the homogeneous ensembles, and is comparable to random forest. This is also true in terms of NLL except the case of random forest, which is performing slightly better. Again, NLL can be improved by adding more features as discussed in the following sub-section.

Table 5. Comparison between Metanet and homogeneous ensembles.

Algorithm	Accuracy	Balanced accuracy	AUC	NLL
AdaBoost	0.647	0.531	0.775	3.258
Bagging	0.794	0.697	0.868	0.775
LogitBoost	0.759	0.698	0.836	8.246
RandF	0.815	0.741	**0.886**	**0.713**
XGBoost	0.807	0.739	0.876	0.843
ROF	0.749	0.721	0.881	1.019
Metanet	**0.831**	**0.756**	**0.881**	0.725

The results show that the proposed heterogeneous ensemble framework, with only five base classifiers, can outperform ensembles with a large number of homogeneous base learners.

5.4.4. Enriching meta-data

The aforementioned experiments have shown that stacked generalization, using a neural network as the meta-classifier can achieve better results over a large set of problems in comparison to established and state-of-the-art ensemble approaches. In the previous experiments, the meta-data was the class probability values of the first-level classifiers, i.e. $\tilde{\boldsymbol{x}}_j^{cp}$. This concatenation of probabilities was slightly lacking in terms of NLL metric compared to some weighted approaches such as RF and CAWPE. It means that these approaches are more confident in their decisions. This could be related to the type and number of features that are used for training the meta-classifier. When using just the outputs of the first-level classifiers by concatenation, the size of the meta-data features is limited to the number of base-classifiers times the number of classes. This could be much less than enough especially when the number of classes is small. However, there are more possibilities for integrating the outputs of the first-level classifiers to prepare the meta-data. Thus, we have decided to enrich the meta-level data with three different approaches, mainly focused on improving the performance of the proposed model on the NLL metric.

To enrich the feature space of the meta-data, we may use the original data along with the class probabilities or class probability averages, as shown in Equation (5). We call these approaches "Class+Base", or "Mean+Base" if the first-level features are added to $\tilde{\boldsymbol{x}}_j^{cp}$ or $\tilde{\boldsymbol{x}}_j^{avg}$, respectively. We have also investigated using the averages along with the concatenated class probabilities ($\tilde{\boldsymbol{x}}_j^{cp+avg}$), which would be called "Mean+Class". Table 6 shows the results of these approaches, averaged on the 121 datasets. In this table, the approach with concatenated class probabilities ($\tilde{\boldsymbol{x}}_j^{cp}$) without any extra data is referred as "Class". As one can see in Table 6, none of the three approaches show any significant improvement over the concatenation approach.

Table 6. Comparing performance of Metanet with different combination functions.

Algorithm	Accuracy	Balanced accuracy	AUC	NLL
Class	0.831	0.756	0.881	0.725
Mean+Base	0.828	0.751	0.872	0.726
Mean+Class	0.830	0.754	0.876	0.740
Class+Base	0.832	0.756	0.876	0.723

In case of "Mean+Class" even, NLL has deteriorated. However, after looking more closely, we have found out that most cases of improvements for these approaches over our original concatenation scheme, happen when the numbers of classes are large and even better when the numbers of features are also small. Inspired by this observation, we have sorted the datasets first by the number of classes in descending order and then by the number of features in ascending order. Let us look at the results for the first nine problems in this sorted list where the number of classes are at least 15 and at most 100. For such problems in which the number of classes is high, it is not easy to achieve good results in case of NLL. However, the proposed approaches yield improvements in these problems as is shown in Table 7.

Table 7. Comparing performance of Metanet with different combination functions on the problems with at least 15 classes.

Algorithm	Accuracy	Balanced accuracy	AUC	NLL
Class	0.768	0.704	0.961	1.347
Mean+Base	0.768	**0.723**	**0.964**	**1.230**
Mean+Class	**0.771**	0.707	0.960	1.284
Class+Base	**0.771**	0.713	0.964	1.234
CAWPE	0.763	0.729	0.963	1.274
RF	0.756	0.716	0.966	1.515

For these same datasets, RF reaches the NLL value 1.515 and CAWPE yields 1.274. Thus, by using the right set of meta-level features, the proposed model can outperform RF and CAWPE, especially in case of NLL. As we have mentioned before, these results show that when we are using only the class probabilities, the number of features is bound to the number of classes, which in many cases is as small as two.

One approach to mitigate this problem is enriching the metadata with the original (base) data. However, in the face of a large number of original features, the small set of class probabilities could be dominated and lose their effect. Therefore, when the number of classes is larger and the number of original features is small the proposed approaches especially "mean + base" and "Class+Base" are more likely to make improvement over the original concatenation scheme. These results also show that the set of meta-level features are very important to the proposed method and can affect its performance. Especially the means of class probabilities over all the base classifiers could be considered as very meaningful features.

5.4.5. Comparing performance on some individual problems

In this section, we depict the results for some of the problems individually. Inspired by the results and observations in the previous experiments, to fairly select a set of representative datasets (problems), we define a clustering task to divide the problems into six exclusive clusters. Then in each cluster of problems, we compute the average classification accuracy, and select the problem with the nearest accuracy to the average as a representative. For the clustering task, each problem is represented by three features; number of instances, number of features and number of classes. The clusters are constructed using the K-means clustering algorithm with $k=15$, after normalizing the feature space. In a post-processing step, the problems in the clusters with less than five members have been reassigned to other clusters. This step reduced the number of clusters into six. The average feature values of each cluster are shown in Table 8 and the representative problems in

Table **9**. The values of our four metrics for the representatives are depicted in Table 10.

Table 8. Average properties in the six clusters of the 121 problems.

Cluster (cl)	Number of features	Number of classes	Number of instances	Cluster size
1	15.7	8.92	1501.67	12
2	7.92	2.94	642.49	49
3	104.14	5.78	1536.93	14
4	26.65	3.0	15638.69	26
5	44.0	54.28	7676.28	7
6	34.92	3.0	1518.15	13

Table 9. Representative problems selected from each cluster.

Cl	Representative problem	Number of features	Number of classes	Number of instances
1	led-display	7	10	1000
2	lenses	4	3	24
3	conn-bench-sonar-mines-rocks	60	2	208
4	Parkinson's	22	2	195
5	plant-margin	64	100	1600
6	annealing	31	5	898

Most of the problems, with large number of classes are placed in the fifth cluster, which is almost identical to the subset of problems mentioned in sub-section 5.4.4.

The problems with the largest numbers of features are placed on the third cluster. Most members of this cluster have two or three classes. Carefully inspecting algorithm performance on these clusters, as shown in Table **10**, confirms our previous observation; when the number of classes is large and the numbers of features are relatively small, as in the fifth cluster, the "Mean+Base" version of our method, outperforms the other models. However, if the number of classes is small and the number of features is large in comparison, we cannot expect "Mean+Base" to yield the same outstanding results. As another example of this, one can refer to the second cluster.

The second-best results for "Mean+Base" are achieved in this cluster which is the largest cluster. In this cluster, "Mean+Base" outperforms the other version of Metanet very clearly. This cluster is the second largest in case of the number of classes. The number of features is not large in comparison to the number of classes, which again provides suitable situations for the "Mean+Base" version of the proposed model to excel.

Table 10. Comparing performance of Metanet with CAWPE on representative problems.

Problem	Accuracy			AUC			Balanced accuracy			NLL		
	Metanet-mean+base	Metanet-class	CAWPE	Metanet-mean+base	Metanet-class	CAWPE	Metanet-mean+base	Metanet-class	CAWPE	Metanet-mean+base	Metanet-class	CAWPE
led-display	**0.729**	0.725	0.718	0.948	0.938	**0.952**	**0.730**	0.727	0.719	1.343	1.478	**1.271**
lenses	**0.797**	0.731	0.776	0.820	0.699	**0.887**	0.692	0.605	**0.765**	1.097	1.334	**0.924**
conn-bench	0.826	**0.846**	0.815	0.881	0.892	**0.894**	0.823	**0.844**	0.811	0.656	0.618	**0.628**
Parkinson's	0.896	**0.927**	0.903	0.925	0.955	0.952	0.819	**0.889**	0.855	0.445	0.352	**0.345**
plant-margin	**0.821**	0.802	0.812	**0.996**	0.991	**0.996**	**0.821**	0.802	0.812	**0.933**	1.210	0.981
annealing	0.929	**0.942**	0.922	0.951	0.963	**0.964**	0.827	**0.862**	0.821	0.381	**0.320**	0.333

In the case of accuracy and balanced accuracy, our models outperform CAWPE in all the examples. However, the CAWPE model performs better than "Mean+Base", in terms of NLL in all the examples except the "plant-margin", which is the representative of the fifth cluster. This cluster contains the problems with the largest number of classes and smaller number of features (in comparison to the number of classes). As we have mentioned above, in such circumstances the "Mean+Base" version of our model exhibits its best performance.

5.4.6. Statistical significance

In this section, we investigate if the superiority of the proposed model over the other methods is significant. We conducted a nonparametric and paired statistical test over the accuracy differences of the proposed model and the other mentioned heterogeneous models over the 121 datasets. We have used the Wilcoxon's signed-rank test for matched pairs. In this test, no specific distribution is assumed over the data. The null hypothesis states that the median difference is zero, while the alternative hypothesis (two-sided) states that it is not zero. If the p-value is small then the null hypothesis can be rejected at the confidence level of 5%, in favour of the alternative. It means that the differences are significant. The test is performed by calculating the signed differences of each pair of the data. Then a statistic is obtained using the rank (signed) of the absolute values of these differences. Our results show that the differences are significant with the p-values less than 0.001.

6. Conclusion

Classification is still a major concern in computer science as finding the perfect classifier for a problem is time-consuming and difficult. We proposed a method to create a heterogeneous ensemble by combining established and state-of-the-art classifier algorithms and neural networks. The proposed method is called Metanet. This model follows the stacking scheme, and uses a neural network as the meta-classifier to learn a nonlinear function for effectively combining the outputs of the base classifiers. The base outputs are used to generate a set of meta-data as input of the meta-classifier. Different functions are proposed to combine the outputs of the base classifiers into meta-data, based on concatenation and averaging. As the number of such features are small, we also considered enriching them with the base-level data.

A set of extensive experiments were conducted to evaluate Metanet with the two mentioned combination functions in comparison to classifiers and ensembles of several different schemes such as homogeneous, weighting scheme, and stacking methods. According to our experiments, Metanet is more accurate compared to other state-of-the-art heterogeneous or homogenous ensembles of weighting or stacking schemes such as CAWPE or XGBoost. Between the two meta-data functions, i.e. concatenation and averaging, averaging performs better when used along with the first-level data. Using a neural network as a combiner in heterogeneous ensembles has the benefits of learning a powerful combination specific to each problem based on the provided data. Our experiments show that a neural network in a stacking scheme, given a sufficient set of meta-data would create a very successful ensemble of heterogeneous classifiers. Further investigation of the combination functions would be a good path to create even more powerful ensembles. Especially with the massive datasets available for many problems, and the continuous improvement in access to computational power, the capability of different ensemble structures should be revisited.

References

[1] W. J. Tastle, *Data mining applications using artificial adaptive systems*, Springer Science & Business Media,. 2013.

[2] J. Large, J. Lines, and A. Bagnall, "A probabilistic classifier ensemble weighting scheme based on cross-validated accuracy estimates," *Data Min. Knowl. Discov.*, vol. 33, no. 6, pp. 1674–1709, 2019.

[3] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 226–239, Mar. 1998.

[4] L. I. Kuncheva and J. J. Rodríguez, "A weighted voting framework for classifiers ensembles," *Knowl. Inf. Syst.*, vol. 38, no. 2, pp. 259–275, 2014.

[5] L. I. Kuncheva, "Switching between selection and fusion in combining classifiers: An experiment," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 32, no. 2, pp. 146–156, 2002.

[6] H. R. Kadkhodaei, AME Moghadam, and M. Dehghan, "HBoost: A heterogeneous ensemble classifier based on the Boosting method and entropy measurement", *Expert Systems with Applications* vol. 157:113482, 2020.

[7] E. Soltanmohammadi, M. Naraghi-Pour, and M. van der Schaar, "Context-based unsupervised ensemble learning and feature ranking", *Machine Learning* vol. 105, pp. 459–485, 2016.

[8] F. Pinagé, dos EM. Santos, and J. Gama, "A drift detection method based on dynamic classifier selection", *Data Mining and Knowledge Discovery* vol. 34, pp. 50–74, 2020.

[9] T. T. Nguyen, N. Van Pham, M. T. Dang, A. V. Luong, J. McCall, and A. W. C. Liew, "Multi-layer heterogeneous ensemble with classifier and feature selection," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, Jun. 2020, vol. 100, pp. 725–733.

[10] K. Zhao, T. Matsukawa, E. Suzuki, "Experimental validation for N-ary error correcting output codes for ensemble learning of deep neural networks", *Journal of Intelligent Information Systems* vol. 52, pp.367–392, 2019.

[11] D. Jimenez, "Dynamically weighted ensemble neural networks for classification," in *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)*, 1998.

[12] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *J. Mach. Learn. Res.*, vol. 8, pp. 2755–2790, 2007.

[13] Y. Zhang, G. Cao, B. Wang, and X. Li, "A novel ensemble method for k-nearest neighbor," *Pattern Recognit.*, vol. 85, pp. 13–25, 2019.

[14] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, Jan. 1992.

[15] J. Kittler and F. M. Alkoot, "Sum versus vote fusion in multiple classifier systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 1, pp. 110–115, 2003.

[16] P. K. Chan and S. J. Stolfo, "Learning Arbiter and Combiner Trees from Partitioned Data for Scaling Machine Learning," in *KDD95*, 1995, pp. 39–44. [Online]. Available: https://www.aaai.org/Papers/KDD/1995/KDD95-047.pdf

[17] P. Chan and S. Stolfo, "Toward parallel and distributed learning by meta-learning," in *Working notes of the AAAAI-93 workshop on Knowledge Discovery in Databases*, 1993, pp. 227–240. [Online]. Available: https://www.aaai.org/Papers/Workshops/1993/WS-93-02/WS93-02-020.pdf

[18] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born Again Neural Networks," in *35th International Conference on Machine Learning, ICML 2018*, May 2018, vol. 4, pp. 2615–2624. [Online]. Available: http://arxiv.org/abs/1805.04770

[19] A. Dutt, D. Pellerin, and G. Quénot, "Coupled ensembles of neural networks," *Neurocomputing*, vol. 396, pp. 346–357, Jul. 2020.

[20] A. Petrakova, M. Affenzeller, and G. Merkurjeva, "Heterogeneous versus Homogeneous Machine Learning Ensembles," *Inf. Technol. Manag. Sci.*, vol. 18, no. 1, 2016.

[21] M. P. Sesmero, A. I. Ledezma, and A. Sanchis, "Generating ensembles of heterogeneous classifiers using Stacked Generalization," *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 5, no. 1, pp. 21–34, 2015.

[22] K. M. Ting and I. H. Witten, "Issues in Stacked GeneralizationS," *J. Artif. Intell. Res.*, vol. 10, pp. 271–289, 1999.

[23] T. K. Ho, "The Random Subspace Method for Constructing Decision Forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, 1998.

[24] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, vol. 13-17-Augu, pp. 785–794.

[25] C. E. Brodley and T. Lane, "Creating and exploiting coverage and diversity," *Proc. AAAI Work. Integr. Mult. Learn. Model.*, pp. 8–14, 1996, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.51.2526&rep=rep1&type=pdf

[26] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996, doi: 10.1007/BF00058655.

[27] H. Drucker, "Improving regressors using boosting techniques," in *14th International Conference on Machine Learning*, 1997, pp. 107–115. [Online]. Available: http://www.researchgate.net/publication/2424244_Improving_Regressors_using_Boosting_Techniques/file/3d

eec51ae736538cec.pdf%5Cnhttp://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.314

[28] Sh. Kashef, H. Nezamabadi-pour, "MLIFT: Enhancing Multi-label Classifier with Ensemble Feature Selection", *Journal of AI and Data Mining*, vol. 7, no. 3, pp. 355-365, 2019.

[29] L. I. Kuncheva and E. Alpaydin, *Combining Pattern Classifiers: Methods and Algorithms*, vol. 18, no. 3. 2007.

[30] S. B. Oh, "On the relationship between majority vote accuracy and dependency in multiple classifier systems," *Pattern Recognit. Lett.*, vol. 24, no. 1–3, pp. 359–363, 2003.

[31] R. M. O. Cruz, R. Sabourin, and G. D. C. Cavalcanti, "On meta-learning for dynamic ensemble selection," in *Proceedings - International Conference on Pattern Recognition*, 2014, pp. 1230–1235.

[32] T. Zhang and G. Chi, "A heterogeneous ensemble credit scoring model based on adaptive classifier selection: An application on imbalanced data," *Int. J. Financ. Econ.*, no. August, p. ijfe.2019, Aug. 2020.

[33] M. Smętek and B. Trawiński, "Selection of heterogeneous fuzzy model ensembles using self-adaptive genetic algorithms," *New Gener. Comput.*, vol. 29, no. 3, pp. 309–327, 2011.

[34] E. Menahem, L. Rokach, and Y. Elovici, "Combining one-class classifiers via meta learning," in *International Conference on Information and Knowledge Management, Proceedings*, 2013, pp. 2435–2440.

[35] G. Tsoumakas, L. Angelis, and I. Vlahavas, "Selective fusion of heterogeneous classifiers," *Intell. Data Anal.*, vol. 9, no. 6, pp. 511–525, 2005.

[36] S. Džeroski and B. Ženko, "Is combining classifiers with stacking better than selecting the best one?," *Mach. Learn.*, vol. 54, no. 3, pp. 255–273, 2004.

[37] T. T. Nguyen, A. V. Luong, M. T. Dang, A. W. C.Liew, and J. McCall, "Ensemble selection based on classifier prediction confidence", *Pattern Recognition*, vol. 100, 107104, 2020.

[38] R. Caruana, A. Munson and A. Niculescu-Mizil, "Getting the Most Out of Ensemble Selection," *Sixth International Conference on Data Mining (ICDM'06)*, Hong Kong, China, 2006, pp. 828-833.

[39] A. M. Webb *et al.*, "To Ensemble or Not Ensemble: When does End-To-End Training Fail?," in *In Computer Vision and Pattern Recognition (CVPR)*, Feb. 2019, pp. 1–21.

[40] L. Rokach, "Ensemble Methods for Classifiers," in *Data Mining and Knowledge Discovery Handbook*, no. August, New York: Springer-Verlag, 2015, pp. 957–980.

[41] Y. Baghoussi and J. Mendes-Moreira, "Instance-Based Stacked Generalization for Transfer Learning," in *Intelligent Data Engineering and Automated Learning*, 2018, pp. 753–760.

[42] A. K. Seewald and J. Fürnkranz, "An evaluation of grading classifiers," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2189, pp. 115–124, 2001.

[43] A. K. Seewald. and J. Fürnkranz, "Grading classifiers," Austrian Research Institute for Artificial Intelligence, Vienna. Tech. Rep. OEFAI-TR-2001-01, 2001.

[44] A. K. Seewald, "Towards a Theoretical Framework for Ensemble Classification (extended version)," Austrian Research Institute for Artificial Intelligence, Vienna, Tech. Rep. TR-2003-08, 2003.

[45] D. V. Sridhar, R. C. Seagrave, and E. B. Bartlett, "Process Modeling Using Stacked Neural Networks," *AIChE J.*, vol. 42, no. 9, pp. 2529–2539, 1996.

[46] A. Ledezma, R. Aler, A. Sanchis, and D. Borrajo, "GA-stacking: Evolutionary stacked generalization," *Intell. Data Anal.*, vol. 14, no. 1, pp. 89–119, 2010.

[47] P. Shunmugapriya and S. Kanmani, "Optimization of stacking ensemble configurations through Artificial Bee Colony algorithm," *Swarm Evol. Comput.*, vol. 12, pp. 24–32, 2013.

[48] N. Rooney, D. Patterson, and C. Nugent, "Non-strict heterogeneous Stacking," *Pattern Recognit. Lett.*, vol. 28, no. 9, pp. 1050–1061, 2007.

[49] Y. Xia, C. Liu, B. Da, and F. Xie, *A novel heterogeneous ensemble credit scoring model based on bstacking approach*, vol. 93. Elsevier Ltd, 2018.

[50] M. Massaoudi, S. S. Refaat, I. Chihi, M. Trabelsi, F. S. Oueslati, and H. Abu-Rub, "A novel stacked generalization ensemble-based hybrid LGBM-XGB-MLP model for Short-Term Load Forecasting," *Energy*, vol. 214, p. 118874, 2021.

[51] J. Yan and S. Han, "Classifying Imbalanced Data Sets by a Novel RE-Sample and Cost-Sensitive Stacked Generalization Method," *Math. Probl. Eng.*, vol. 2018, 2018.

[52] S. Rajaraman *et al.*, "A novel stacked generalization of models for improved TB detection in chest radiographs," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, 2018, vol. 2018-July, pp. 718–721.

[53] Z. Eivazpour and M. R. Keyvanpour, "CSSG: A cost-sensitive stacked generalization approach for software defect prediction," *Softw. Test. Verif. Reliab.*, vol. 31, no. 5, 2021.

[54] K. Akyol, "Stacking ensemble based deep neural networks modeling for effective epileptic seizure detection," *Expert Syst. Appl.*, vol. 148.

[55] A. Das, S. Roy, U. Bhattacharya, and S. K. Parui, "Document Image Classification with Intra-Domain Transfer Learning and Stacked Generalization of Deep

Convolutional Neural Networks," *Proc. - Int. Conf. Pattern Recognit.*, vol. 2018-Augus, pp. 3180–3185, 2018.

[56] J. C. Platt, "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods," *Adv. large margin Classif.*, vol. 10, no. 3, pp. 61–74, 1999.

[57] P. Domingos and F. Provost, "Tree Induction for Probability-Based Ranking," *Mach. Learn.*, vol. 52, no. 3, pp. 199–215, 2003.

[58] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," in *Twenty-first international conference on Machine learning - ICML '04*, 2004, no. 1996, p. 18.

[59] A. K. Seewald and J. Fuernkranz, "An evaluation of grading classifiers," in *International Conference on Advances in Intelligent Data Analysis, Proceedings*, 2001, pp. 115-124.

[60] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996, [Online]. Available: https://www.stat.berkeley.edu/%7B~%7Dbreiman/bagging.pdf

[61] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.

[62] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors)," *Ann. Stat.*, vol. 28, no. 2, pp. 337–407, 2000.

[63] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso, "Rotation Forest: A new classifier ensemble method". *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 10, pp. 1619-1630, 2006.

یادگیری ترکیب دسته‌بندهای تعمیم‌یافته ناهمگن با استفاده از شبکه‌های مصنوعی عصبی

مرضیه رحیمی*، امیرعلی طاهری و هدی مشایخی

دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شاهرود، شاهرود، سمنان، ایران.

ارسال ۲۰۲۲/۱۱/۰۹؛ بازنگری ۲۰۲۲/۱۱/۲۵؛ پذیرش ۲۰۲۲/۱۲/۲۱

چکیده:

یافتن راهی موثر برای ترکیب یادگیران پایه، بخشی اساسی از ساخت یک سیستم یادگیری ناهمگن گروهی است. در این مقاله، یک چهارچوب کلی برای یادگیری ناهمگن گروهی ارائه می‌کنیم که در آن از یک شبکه مصنوعی عصبی، برای یادگیری ترکیب غیرخطی دسته‌بندهای پایه استفاده می‌شود. در چهارچوب پیشنهادی، مجموعه‌ای از دسته‌بندهای ناهمگن در کنار یکدیگر و در طی چند مرحله به تولید خروجی مرحله اول می‌پردازند. به این ترتیب که در هر مرحله بخشی از خروجی، به شیوه‌ای مشابه اعتبارسنجی متقابل، تولید می‌شود. این خروجی‌ها با استفاده از چندین تابع مختلف غنی و ترکیب شده و ورودیهای دسته‌بند مرحله بعد را تولید می‌کنند. ما آزمایشات گسترده‌ای را بر روی ۱۲۱ مجموعه داده انجام داده و روش پیشنهادی را با دیگر روش‌های ناهمگن موجود اعم از نوین و شناخته‌شده، مقایسه نموده‌ایم. نتایج این آزمایشات حاکی از برتری روش پیشنهادی بر بسیاری از روش‌های ناهمگن موجود و همینطور دسته‌بندهای منفرد با تنظیم دقیق است. روش پشنهادی، همچنین با تعدادی روش‌های همگن مقایسه شده و برتری خود را نشان داده است. یافته‌های ما نشان می‌دهد که برتری روش پیشنهادی بر روی مجموعه داده‌های بزرگ چشمگیرتر است.

کلمات کلیدی: یادگیری ناهمگن گروهی، دسته‌بندی، شبکه‌های عصبی، تعمیم انباشته، همجوشی دسته‌بندها، یادگیری ماشین.