Information Fusion 31 (2016) 26–42

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Improved classification with allocation method and multiple classifiers

Sašo Karakatič*, Vili Podgorelec

Faculty of Electrical Engineering and Computer Science, Institute of Informatics, University of Maribor, Slovenia

ARTICLE INFO

Article history:
Received 14 May 2015
Revised 15 December 2015
Accepted 19 December 2015
Available online 29 December 2015

Keywords:
Classification
Ensemble methods
Allocation
Anomaly detection

ABSTRACT

Classification is the most used supervized machine learning method. As each of the many existing classification algorithms can perform poorly on some data, different attempts have arisen to improve the original algorithms by combining them. Some of the best know results are produced by ensemble methods, like bagging or boosting. We developed a new ensemble method called allocation. Allocation method uses the allocator, an algorithm that separates the data instances based on anomaly detection and allocates them to one of the micro classifiers, built with the existing classification algorithms on a subset of training data. The outputs of micro classifiers are then fused together into one final classification. Our goal was to improve the results of original classifiers with this new allocation method and to compare the classification results with existing ensemble methods. The allocation method was tested on 30 benchmark datasets and was used with six well known basic classification algorithms (J48, NaiveBayes, IBk, SMO, OneR and NBTree). The obtained results were compared to those of the basic classifiers as well as other ensemble methods (bagging, MultiBoost and AdaBoost). Results show that our allocation method is superior to basic classifiers and also to tested ensembles in classification accuracy and *f*-score. The conducted statistical analysis, when all of the used classification algorithms are considered, confirmed that our allocation method performs significantly better both in classification accuracy and *f*-score. Although the differences are not significant for each of the used basic classifier alone, the allocation method achieved the biggest improvements on all six basic classification algorithms. In this manner, allocation method proved to be a competitive ensemble method for classification that can be used with various classification algorithms and can possibly outperform other ensembles on different types of data.

1. Introduction

Classification is a technique where an algorithm learns a classification model on a given dataset of labeled instances of how to predict the labels (or classes) of new, unseen (not yet labeled) instances. Many different classification approaches have been proposed and used for solving real life problems, ranging from statistical methods to machine learning techniques as linear classifiers (Naive Bayes classifier and logistic regression), distance estimations (*k*-nearest neighbours), support vector machines, rule and decision tree based methods, and the neural networks, to name a few. The prediction performance of these basic methods can be improved significantly with the use of ensembles – a committee of models, where several models collaborate in classification process with the goal of more stable and more accurate classification results. This process is called *ensemble learning* and has been studied extensively [5,8,30] and [36] with both empirical [18,30,33] and theoretical [16,28] evidence that combination of classifiers can produce superior results in comparison to basic classifiers alone. Most used and more popular ensemble methods are following: boosting methods AdaBoost [11] and its extension MultiBoost [48], bagging [3], arbitrating [34], stacked generalization [49], delegation [9], averaged decision trees [23] and weighting of individual classifiers [46]. The nature of such methods is that the final classifier is combined from multiple basic classifiers, where each one of them is constructed on specific settings or subset of instances and none of them has a full set of information about the problem.

We propose a novel ensemble method of *allocation* where multiple micro classifiers are used and together they form a large macro classifier. As with previously mentioned ensemble methods, where numerous classifiers get each of their own set of instances to be classified, here also each micro classifier gets its set of instances, but in contrast to other methods – sets of instances are not chosen at random and sets do not overlap over each other. In the beginning of proposed allocation technique, the allocation method splits an original dataset of instances and allocates each of these sets to one specialized classifier, which is trained to deal with such instances. As opposed to delegation method where the classifier alone determines when it is no longer capable of classification of particular instance and thus delegates them to another

* Corresponding author. Tel.: +386 31869109.
E-mail address: karakatic@gmail.com, saso.karakatic@um.si (S. Karakatič).

http://dx.doi.org/10.1016/j.inffus.2015.12.006
1566-2535/

classifier, our method eliminates this need where classifiers must be modified so that they can learn when to delegate. Instead, it introduces the allocator that reviews instances and allocates them to one of the specialized micro classifiers. Each micro classifier works independently from the allocator and other micro classifiers and do not need any modification – any existing classification algorithm can be used. Allocation method is thus a parallel method, where no sampling is done and can efficiently deal with bi-class and multi-class instances. The type of classifier plays no role in this process as will be shown later when we analyze allocation method on several classification algorithms. The key point in the process is the splitting of original data in adequate sets. In this paper we propose the allocation method with two micro classifiers and the allocator based on one-class support vector machine method (SVM) used as anomaly detection method. In general, other methods and arrangements (more than two micro classifiers) could be used for allocation, but were not tested and are out of scope of this paper.

Related methods have been reviewed in [15] and [50], ranging from the gating networks where mixed expert method is used based on the probabilities of density estimations [21] to mixtures of experts used with the expectation maximization algorithm [24]. In [14] researchers used the subspace projections to construct ensembles of classifiers. Similar research was done in [47] where researchers used the partitioning of the features to form different subspaces, which were used in training of ensembles. In contrast to our method, these methods use some kind of randomized splitting of the initial dataset, whereas our allocation method uses classification method (in the case of this paper the one-class SVM is used) for non-random splitting of the instances and therefore enabling micro classifiers to work with specific type of data. In [27] authors use one-class classification, which is also used as the allocator in our approach, but they combine multiple one-class models to classify the instances. Allocation method uses the one-class anomaly detection classification only to divide the initial dataset and continues to use other classification algorithms for inducing micro classifiers. One-class SVM method, similar to our implementation of allocation method, is used in anomaly detection system [45], where the authors used multiple SVM models to remove normal data and identify the outliers.

A number of improvements in boosting algorithms have been made, from previously mentioned AdaBoost and MultiBoost to other boosting methods where some kind of oversampling algorithm is used. RUSBoost [40] and its newer variant EUSBoost [13] use oversampling in order to optimize performance on skewed and highly imbalanced datasets. Allocation method does not use any over- or under-sampling and so does not reshape the data with artificial data instances to construct the classification model. Besides sampling, allocation method is not an iterative process, where classification methods are improved through steps. Micro classifiers in allocation method work independently of each other and do not vote for final classification decision; they rather specialize in one type of instances and full trust is given in their sole decision. The biggest difference of allocation method with regard to ensemble methods is in the fact, that each new instance is always classified only by one micro classifier, determined by allocation model. In this regard, the allocation method might not be considered as a classical ensemble (where each classifier contributes to decision on each instance), but rather as a specialization (where each classifier is solely responsible for its own subset of instances).

The contributions of this work are as follows:

- We propose a new allocation method for building a simple ensemble of two basic classifiers and one allocator. The allocator is based on one-class SVM for anomaly detection; it splits an original dataset into two disjoint subsets, of which each one is used to train one micro classifier using any of the existing classification algorithms.
- The allocation method proposes an easy-to-use approach, which does not require any further configuration or parameter setting from the user and works on any classification dataset. The allocation method thus ensures a simple and universal, yet competitive and scalable classification approach which provides very good classification performance.
- We perform extensive computational tests on a diverse set of benchmark datasets that demonstrate the strength of the proposed allocation method and show that it outperforms basic classifiers as well as standard classification ensemble methods.
- We discuss the possibility of extending our allocation method using different allocation approaches and algorithms, combinations of micro classifiers, and more than two micro classifiers in the ensemble.

The goal of our research was to test whether our technique of allocation is a valid method for solving classification problems and how it performs on different datasets and using different classification algorithms. Although partitioning of a dataset is quite a standard strategy for improving results on almost any specific domain, we wanted to show how the proposed allocation approach, which is a universal method independent of a domain, performs in general over multiple datasets from several domains and using several basic classifiers. The experiment was conducted where allocation method was tested on 30 standard benchmark datasets with the use of six well known classification algorithms used to build the micro classifiers. Results are compared to other popular ensemble techniques.

The remaining of this paper is organized as follows. Second chapter starts with review of similar methods where we compare the allocation method with the existing research body of ensemble methods. Next is the description of one-class SVM for anomaly detection, which is an important part of allocation method. After that the description and basic formulation of allocation method is presented. In third chapter we describe the layout of the experiment and present the results supplemented with the statistical analysis, and discussion. In the end we conclude with final remarks about allocation method and results of the experiment, and propose a development for future research.

2. Proposed improved classification with allocation method

Many researches indicate that removing outlier instances before the process of classification can significantly improve the classification model. The process of outlier removal was done in combination with traditional classification algorithms to achieve better classification accuracy [41]. Identifying and removing potentially problematic instances has also been done with PRISM filtering method and produced statistically significant improvements in classification metrics [43]. In another research authors used only particular data in classification process to improve classification of music genres [29]. In [44] authors used instance based learning algorithms and proved that datasets reduced of noise can lead to better classification models [44]. Recent survey additionally confirms that classification metrics suffer with the presence of noise in the dataset and so data instances should be chosen to produce the best performing models [10]. Another review overviews the instance selection methods and their contribution to the classification metrics improvements [32]. The improvements that these outliers removal methods made makes a case for their usage in the classification process.

Few researches take this approach to another level, with the theory that too large datasets do not perform well in construction of classification models, mainly for the same reasons as

before – noise and anomalies. In [17] authors introduce the divide and conquer approach where they split dataset in recursive manner into smaller sets all the way down to appropriately sized ones that are used in final classification process. Similar process was performed with the help of genetic algorithms, where the splitting decision tree was constructed in evolutionary process [35]. Another research was published that uses the concept of dividing the datasets and using smaller sets for more accurate classification [51]. Here the researchers used the boosting process on multiple weak classifiers but on split datasets to achieve better results. All these researches show that, again as with outlier removal, classification can be done in a more accurate manner if datasets tend to be smaller and without distractions (outliers).

We build on both presented ideas – using outlier removal and divide and conquer approach simultaneously – to provide classification methods with as homogeneous data as possible, but to retain all of the instances and not losing any information. Unlike outlier removal methods, where outliers are removed completely from the classification process, our allocation method just separates potential outliers, but still sends them through the regular process of model construction to preserve all the information of the dataset, which is especially important in the case of heavily skewed or small datasets. In contrast to divide and conquer algorithms, we do not split dataset with regular process of classification, but we use anomaly detection method that is trained only on majority data (which is considered as normal in this case) and splits the original dataset into two subsets – "normal" and "anomaly" datasets. This insures that "normal set" is as homogeneous as possible, without the anomalies that can worsen the classification models, as seen in previously mentioned researches. On the other hand, the "anomaly set" is mainly without the majority labeled data and so there is no more majority pressure in training the classifiers on it, and thus the recognition of instances of under-represented classes are more easily identified.

2.1. Allocating data with one-class SVM anomaly detection

In this paper, for the anomaly detection algorithm, we use one class support vector machine (SVM) algorithm which was first proposed by Schölkopf et al. in [39] and expanded the original SVM formulation by Cortes and Vapnik [6]. For a good review about SVM methodologies see paper [42]. The idea with one-class SVM is to teach the algorithm what the normal data looks like and to determine how far from the normal set of data is still considered normal. All instances outside of this normal buffer area are labeled as anomalies. Let us consider the training instances

$$x_1,\ x_2, \ldots, x_n \in \mathrm{X} \tag{1}$$

from set X that is contained in the feature space F where $n \in \mathbb{N}$ is the size of the X. The goal of the one-class SVM is to find the hyperplane, from feature space F that contains the normal data and represents the buffer zone for normality. As the hyperplane sometimes cannot be found with the use of feature space F, we use the feature map Φ that maps the set X to richer feature space F'. From numerous kernel functions available, we used the Gaussian radial basis function (RBF) for kernel function K which is formulated as:

$$K(x_i, x) = e^{-\gamma \|x_i - x\|^2} \tag{2}$$

where parameter $\gamma > 0$ is fixed. After the inputs are mapped into a new feature space F', the hyperplane parameters are determined by solving the following quadratic programming problem:

$$\min_{w,\xi,\rho} \left(\frac{1}{2}\|w\|^2 + \frac{1}{C\,n}\sum_i \xi_i - \rho \right) \tag{3}$$

subject to the inequality constraints

Algorithm 1: GetMajorityClasses

```
Data: classification dataset X
Result: array of indexes of majority classes as majority_classes
begin
    num_of_classes ← NumberOfClasses(X)
    class_ratios ← DistributionOfClasses(X)
    half ← num_of_classes / 2

    if isOdd(num_of_classes) then
        half ← half + 1
    end

    majority_classes ← [ ]

    foreach i ∈ [1, half] do
        largest_class_i ← GetLargestClass(class_ratios)
        majority_classes[i] ← largest_class_i
        RemoveLargestClass(largest_class_i)
    end
end
```

Fig. 1. Pseudocode that returns the majority classes from the data.

$$w \cdot \Phi(x_i) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad \forall i = 1, 2, \ldots, \ n \tag{4}$$

where C is the fraction of instances that are ignored by the hyperplane and can land on the wrong side of the hyperplane, w is the vector that is orthogonal to the separating hyperplane, ξ is a vector of slack variables that penalizes the rejected hyperplane patterns, and ρ is the margin of the hyperplane to the nearest not-ignored instance. The SVM function is defined by equation:

$$f(x) = sign\left(\sum_{i=1}^{n} \alpha_i K(x_i, x) \geq \rho \right) \tag{5}$$

Here *sign* is the function that returns +1 for a positive input and 0 otherwise, vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ consists of the coefficients and $\rho > 0$ is the margin.

2.2. Building and using the two classifiers

The anomaly detection algorithm splits data to "normal" instances and "anomalies" and then forwards them each to their corresponding micro classifier. Let us look into the process of how anomaly detection model and each of the micro classifiers are made (Fig. 2). As we start with all of the training data intact, we first have to identify what are the majority data and which are the minority data. The algorithm shown in Fig. 1 calculates the distribution of the classes in the dataset and returns the labels of the majority classes.

As is shown in the pseudo code (see Fig. 1), the majority consists of the half (or half + 1, in case of odd number of class labels) of the class labels that represent the largest portion of the whole dataset. Following always applies:

$$\begin{aligned} &\mathrm{X}_1 \cup \mathrm{X}_2 \cup \ldots \cup \mathrm{X}_m = \mathrm{X} \\ &\mathrm{X}_i \cap \mathrm{X}_j = \emptyset, \quad \forall i = 1, 2, \ldots, m, \quad \forall j = 1, 2, \ldots, m, \quad i \neq j \\ &\mathrm{X}_i \subseteq \mathrm{X}, \quad \forall \mathrm{i} = 1, 2, \ldots, m \end{aligned} \tag{6}$$

where X_i is set of instances of the same class, X is the whole dataset and m is the number of all classes in the whole dataset. As we split the original dataset in the majority and minority data sets, this applies:

$$\begin{aligned} &\Upsilon_{Ma} \cup \Upsilon_{Mi} = X, \quad \Upsilon_{Ma} \cap \Upsilon_{Mi} = \emptyset \\ &X_1 \cup X_2 \cup \ldots \cup X_{mN} = \Upsilon_{Ma}, \quad X_{mN+1} \cup X_{mN+2} \cup \ldots \cup X_m = \Upsilon_{Mi} \\ &mN = \left\lceil \frac{m}{2} \right\rceil, \quad mN \geq m - mN, \quad |\Upsilon_{Ma}| \geq |\Upsilon_{Mi}| \end{aligned} \tag{7}$$

Y_{Ma} is the majority dataset, Y_{Mi} is the minority dataset, mN is the number of majority classes and m-mN is the number of minority classes.

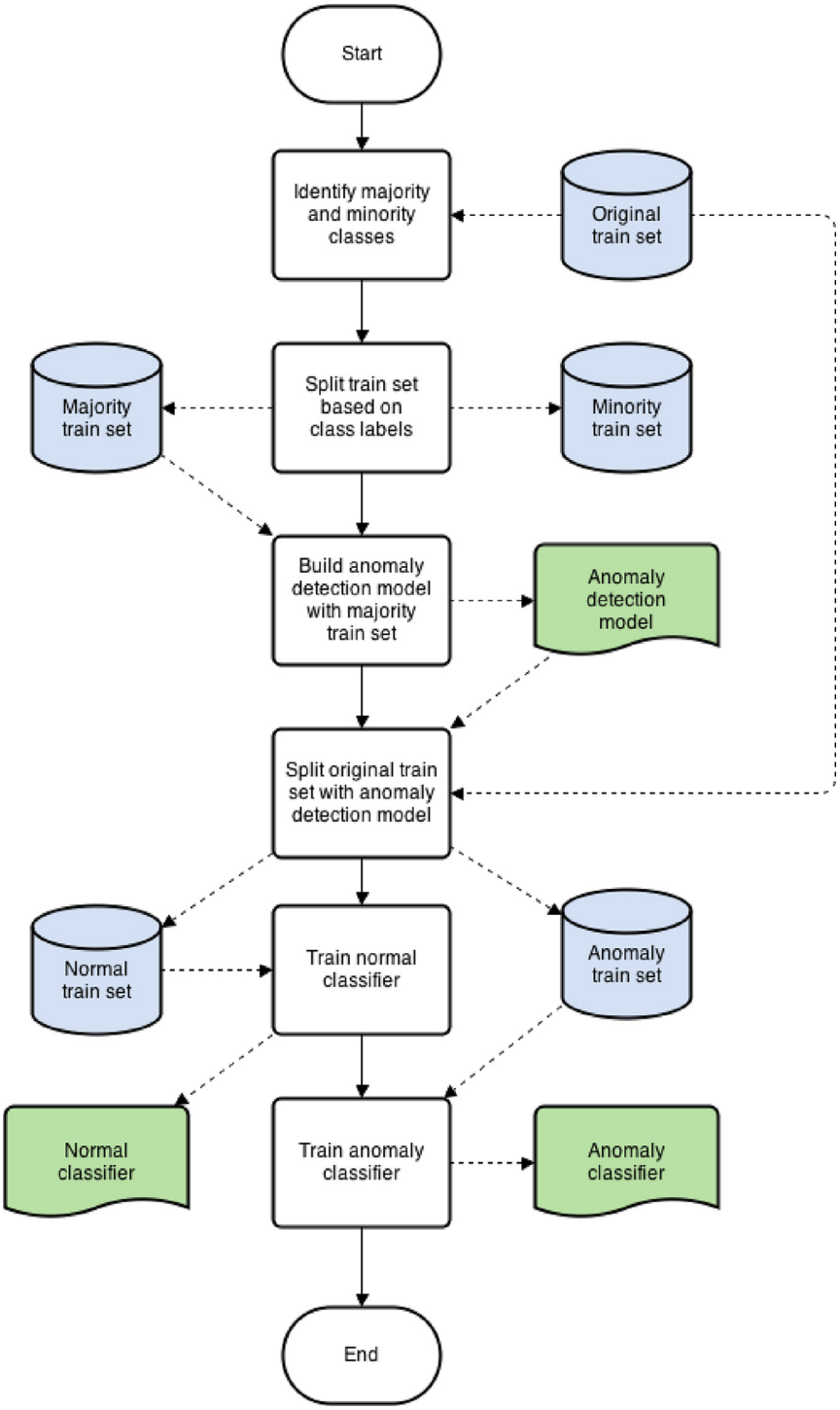


Fig. 2. The flow diagram for training of the anomaly detection model and the classifiers.

The first step is to identify which classes represent the majority and which represent the minority instances. Algorithm returns the array of the majority class labels which is used for filtering in the next step in the Fig. 2 where we use only the instances that represent one of the majority classes and put together the majority training set. With this majority training set we train the anomaly detection model that can then effectively identify normal and anomaly instances. The training of the anomaly detection model requires input of only "normal" data (majority set in our case), so that it can learn the properties of the normal instances. All instances, which are later not recognized as being compliant enough with these properties, are regarded then as anomalies. Original data (before majority-minority split) is then evaluated by anomaly detection model and is split into two groups – normal and anomaly datasets. This split is generally not the same as before with the majority and minority split. Majority-minority split was done based on the class labels and was used only to train the anomaly detection model about properties of normal instances, but normal-anomaly split is done with the help of anomaly detection model regardless of the class label of the instances. The normal set can be the same as the majority set, but this is not generally the case. Anomaly detection model ignores class labels which is necessary in order to be used with new, unseen instances (the test set), where class label is unknown. In this manner, the unknown

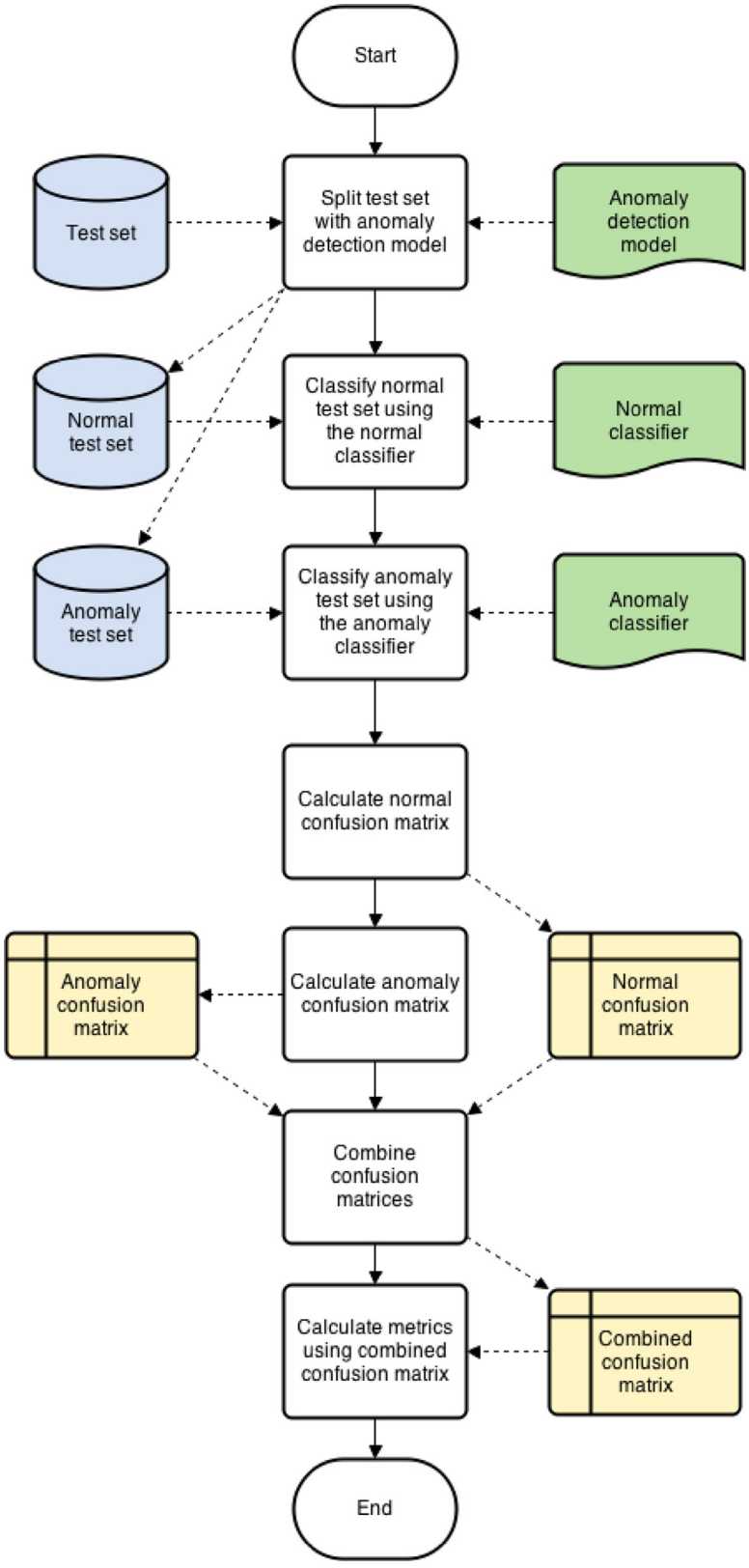


Fig. 3. The flow diagram for using and testing resulting solution with three models.

instances can be allocated to either normal or anomaly (micro) classifier.

Normal and anomaly datasets are then used to train each of their own micro classifier – normal training dataset is used to train normal classifier and anomaly training dataset is used to train anomaly classifier. Resulting solution – the macro classifier, i.e. the ensemble – then consists of three models – anomaly detection model (the allocator), normal classification model (first micro classifier) and anomaly classification model (second micro classifier).

Now let us describe how we can use and test this resulting ensemble of three models (Fig. 3). First the test set is split with the anomaly detection model into two test sets – the normal test set

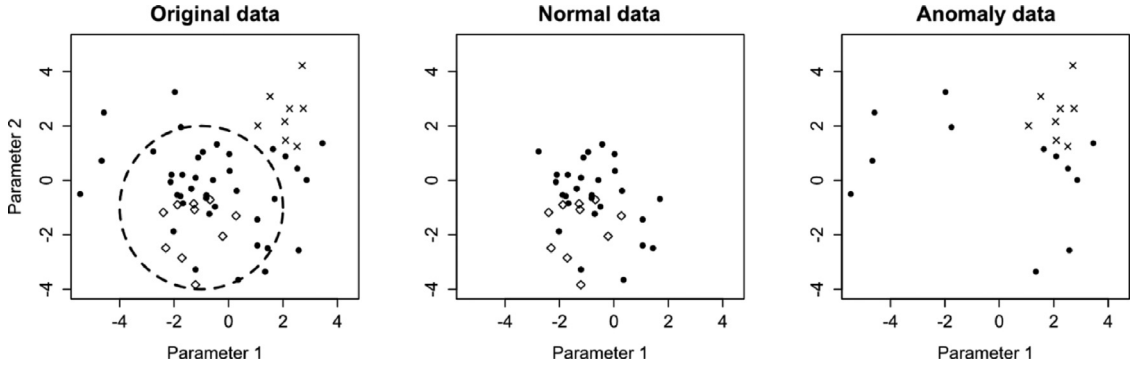


Fig. 4. An example of anomaly detection model: original data with split boundary, and the normal and anomaly data.

and the anomaly test set. Here we can see why the anomaly detection model was trained – because we need a way to split the data into normal and anomaly sets when we do not know the class labels of the data. The normal data is then classified with the normal classifier and the anomaly data is classified with the anomaly classifier. Resulting confusion matrices CM_{normal} and $CM_{anomaly}$ are then combined (aggregated) in global confusion matrix CM_{total} as is shown in Eq. (8) below, from where the final metrics are calculated.

$$CM_{total} = CM_{normal} + CM_{anomaly}$$

$$CM_{total} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,m} \end{bmatrix} + \begin{bmatrix} b_{1,1} & \cdots & b_{1,m} \\ \vdots & \ddots & \vdots \\ b_{m,1} & \cdots & b_{m,m} \end{bmatrix} = \begin{bmatrix} a_{1,1}+b_{1,1} & \cdots & a_{1,m}+b_{1,m} \\ \vdots & \ddots & \vdots \\ a_{m,1}+b_{m,1} & \cdots & a_{m,m}+b_{m,m} \end{bmatrix} \quad (8)$$

2.3. The use of anomaly detection model for allocation

The anomaly detection model is built to separate normal instances from anomalies. This is done with the SVM method with the one-class setting and the RBF kernel function. From the training data, used to train the SVM model, all the minority class instances are removed, and all the remaining (majority class) instances are stripped of their class labels – they are all relabeled as "normal". SVM builds a model based on this one-class data set as is shown below on Fig. 4. After that, all original training data (majority and minority class instances) are sent through the model and is separated into two groups – the normal instances and anomalies. These two groups are then used to train the two classification models, where the normal instances are used to train the normal classification model and similarly the anomaly classifier is trained using the instances which were labeled as anomalies. Which classification algorithm is used for either normal and/or anomaly data was not the focus of our research; in general, any existing classification algorithm can be used. Our goal was to test if allocation process can improve overall classification performance independent of the classification algorithm used for constructing micro classifiers. Therefore, various classification algorithms were used in our experiment. Although in our experiment we always used the same classification algorithm for both normal and anomaly classifier, there is no need to do so and different classification algorithms can be used as well.

When it comes to testing and using the allocation method, an incoming new data instance is sent to the starting point – to the allocator, the anomaly detection model. This model evaluates the data instance and either classifies it as normal or anomaly. This determines to which of the two classification models the data instance is allocated to – to normal classifier or to anomaly classifier. The selected classifier then performs the actual classification of the data instance into one of the available classes of the data set.

In the Fig. 4 we can see an example of how data are split by the anomaly detection model which acts as the allocator. The sample data used in this example is randomly generated with only two parameters and three class labels. The dotted circle represents the anomaly detection model with very simple boundary of a circle, where instances inside the circle are treated as normal data while the instances outside the circle boundary are treated as outliers or anomaly data.

3. Experimental framework and results

This section describes the methodology followed in the experimental study of the ensemble methods comparison. We will explain the configuration of the experiment: used data sets and performed measurements, as well as the results: obtained accuracy and *f*-score on test sets by different methods, and the results of specific statistical tests. The methods used in the comparison are the same as listed in the introduction.

3.1. Experimental framework

Experiments were implemented with programming language Java and were performed using the standard Weka library and all classification models were used with their default settings. Proposed method of allocation and regular classification models can perform better with tweaked settings, but we chose not to do that for two reasons, the first being that optimal settings are specific for each problem and the second is that our goal was not to find the optimal setting, but to evaluate our allocation method to regular classification methods and other ensembles. For this purpose, all the parameters were using the same default values for all the experiments performed. Anomaly detection was done with the SVM implementation LibSVM [4] with the one-class settings and of radial-SVM kernel with parameter C (cost) set to default value and parameter *nu* set to 0.25. Parameter *nu* can be further optimized to get the optimal results, but the goal of the experiments was to test the validity of the allocation method and not to optimize the method on each dataset and each classifier.

Performance of the ensemble methods is analyzed by using 30 well known and often-used data sets taken from the UCI Machine Learning Database Repository [2]. The main characteristics of these datasets are summarized in Table 1. For each data set, it shows the number of classes (# classes), the number of attributes (# attributes), the number of instances (# instances), ration between majority and minority class frequencies called *imbalance ratio* (IR) shown below in Eq. (9), and the percentage of instances allocated

Table 1
The data sets used in the experiment are from UCI Machine Learning Repository [2].

Data set	# classes	# attributes	# instances	IR	Split %
Anneal	6	38	898	17.71	0.21
Audiology	24	69	226	10.89	0.74
Autos	7	25	205	7.20	0.77
Balance-scale	3	4	625	11.76	0.94
Breast-cancer	2	9	286	2.36	0.43
Breast-w	2	9	699	1.90	0.74
Car	4	6	1728	11.90	0.94
Colic	2	22	368	1.71	0.73
Credit-a	2	15	690	1.25	0.39
Diabetes	2	8	768	1.87	0.75
Ecoli	8	7	336	10.59	0.95
Eye_movements	3	27	10936	2.81	0.81
Glass	7	9	214	8.73	0.93
Heart-statlog	2	13	270	1.25	0.68
Hypothyroid	4	29	3772	37.89	0.98
Ionosphere	2	34	351	1.79	0.09
Iris	3	4	150	2.06	0.87
Letter	26	16	20000	1.05	0.34
Liver-disorders	2	6	345	1.38	0.71
Monks-problem	2	6	432	1.00	0.54
Mushroom	2	22	8124	1.07	0.37
Optdigits	10	64	5620	1.02	0.32
Pendigits	10	16	10992	1.08	0.44
Primary-tumor	22	17	339	8.16	0.61
Segment	7	19	2310	1.34	0.68
Sick	2	29	3772	15.33	0.96
Sonar	2	60	208	1.14	0.48
Soybean	19	35	683	3.46	0.60
Vehicle	4	18	846	1.06	0.14
Vowel	11	13	990	1.22	0.48

to majority by our proposed method (split %).

$$IR = \frac{majority\ size}{minority\ size} \tag{9}$$

The data sets considered were divided into train and test sets in a ratio of 80:20 using fivefold cross-validation procedure. All the presented results are averages of five folds achieved by a specific method on a test set. Folds were again made by Weka with its stratification method, which splits set in such way that distribution of classes remains the same across all of the folds.

3.2. Results and analysis of accuracy and f-score

Six well known classification algorithms have been used throughout the experiment: J48 [37], Naïve Bayes [22], IBk [1], SMO [25], OneR [20], and NBTree [26]. These six classifiers were used as a basis for the following ensemble methods: MultiBoost, bagging, AdaBoost, and our proposed allocation method. For the sake of comparison, the original classification results, obtained by the six classification algorithms, are presented also. In this manner, five different methods have been compared in the experiment: (original, allocation, MultiBoost, bagging, AdaBoost), each on six basic classifiers. For each of the thirty data sets, for each of the five methods, and for each of the six basic classifiers, the following results have been measured (average of five folds): accuracy, precision, recall, specificity, and *f*-score. However, due to the abundance of results, only the two most informative measures are presented, namely accuracy and *f*-score. According to Demšar [7] accuracy and *f*-score are most used metrics in machine learning papers in recent years. Furthermore, *f*-score (also called f-measure) is the harmonic mean of metrics recall and precision as is shown in Eq. (10) below, which combines different measurements (true positive, false positive and false negative rates) together in a single metric.

$$f-score = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{10}$$

Table 2 shows the accuracy and *f*-score results obtained by five methods over test data using J48 as the basic classifier. The column denoted by *Orig* corresponds to the case in which no meta ensemble method is used. The best case (independently for accuracy and *f*-score) in each data set is remarked in bold. Tables 3 to 7 show the same for the other five basic classifiers (NB, IBk, SMO, OneR, and NBTree). The abbreviation *Alloc* corresponds to our method of allocation with anomaly detection (one-class SVM).

From Tables 2 to 7 we can see that on average our proposed allocation method outperforms all other methods and achieves the best average results both in accuracy and in *f*-score, while at the same time standard deviation of results is the lowest in all cases also. These results are summarized in Tables 8 and 9.

Our allocation method achieved the highest number of best accuracy results on single datasets for all classifiers with the exception of J48, where AdaBoost achieved one best result more (AdaBoost achieved best results on 12 data sets while our allocation method achieved best results on 11 data sets). In the case of NBTree classifier our allocation method achieved best results on 9 data sets (the same number of best results achieved also MultiBoost and bagging). In the case of other four classifiers our allocation method achieved the most wins (best ranks) by far, namely 19 in the case of Naive Bayes and SMO, 24 in the case of IBk and even 25 (out of 30 data sets used) in the case of OneR classifier.

When considering the *f*-score results, our allocation method achieved the highest number of best results for all classifiers. While in the case of NBTree the same number of 9 best results was achieved also by MultiBoost and AdaBoost, for other five classifiers our allocation method achieved the most best results by far, namely 12 in the case of J48, 17 in the case of SMO, 19 in the case of Naive Bayes, 23 in the case of IBk and even 25 (out of 30 data sets used) in the case of OneR classifier.

Fig. 5 illustrates average accuracy and *f*-score results obtained on all 30 used data sets for different classification algorithms; the results obtained on six different classifiers are presented as well as the average accuracy and *f*-score on all classifiers in total. It can be seen that our allocation method achieved both the best average accuracy and the best average *f*-score in the case of all single classifiers and consequently also when the average of all the results (all six classifiers on 30 data sets) are considered.

Fig. 6 illustrates the aggregated accuracy results from all six classifiers. It can be seen that all three existing ensemble methods (MultiBoost, bagging and AdaBoost) scored very similar accuracy results in the case of all six classifiers and improved the results of original classifiers (when no ensemble method is used) by approximately one percent on average. On the other side, it can be also seen that our allocation method outperforms these three ensemble methods in the case of all six classifiers. In this manner, it further improved the results by approximately four percent on average.

Fig. 7 illustrates the aggregated *f*-score results from all six classifiers. It can be seen that all three existing ensemble methods still outperform the original classifiers, but their *f*-score results are less similar than accuracy. Bagging improved the original classification results by approximately one percent on average, MultiBoost by approx. one and half percent on average and AdaBoost by approx. two percent on average. On the other side, our allocation method again outperforms all three ensemble methods in the case of all six classifiers. Its improvement of original classification results is thus more than seven percent on average.

Fig. 8 illustrates the accuracy results as obtained by five methods on different classifiers for all 30 data sets. Again it can be

Table 2
Accuracy and *f*-score results obtained by various ensembles of J48 classifier over test data.

	Accuracy					*f*-Score				
Dataset	**Orig**	**Alloc**	**MB**	**Bag**	**Ada**	**Orig**	**Alloc**	**MB**	**Bag**	**Ada**
Anneal	89.33	90.22	91.56	91.11	**92.89**	55.41	72.07	69.70	67.92	**75.90**
Audiology	78.95	78.95	82.46	78.95	**85.96**	40.38	32.34	45.34	44.37	**50.25**
Autos	78.85	67.31	76.92	**82.69**	75.00	70.21	60.70	69.43	**72.85**	68.23
Balance-scale	81.53	**85.35**	83.44	83.44	82.80	56.75	**82.41**	59.03	58.71	62.03
Breast-cancer	**73.61**	70.83	68.06	72.22	72.22	58.98	**63.08**	59.56	59.82	62.96
Breast-w	93.71	**97.71**	96.57	93.71	97.14	93.11	**97.44**	96.23	93.05	96.84
Car	89.35	90.28	**94.21**	89.58	92.13	74.36	81.46	82.51	72.83	**82.62**
Colic	75.00	78.26	78.26	78.26	**81.52**	72.64	74.82	75.67	76.37	**79.48**
Credit-a	87.86	85.55	80.92	**88.44**	84.97	87.73	85.27	80.83	**88.35**	84.82
Diabetes	77.60	**91.15**	77.08	81.77	75.52	75.90	**90.22**	74.22	78.88	72.57
Ecoli	79.76	80.95	82.14	**84.52**	79.76	50.26	51.69	53.42	**57.22**	48.88
Eye_movements	64.52	**71.69**	69.35	68.73	68.62	65.01	**73.59**	70.21	69.48	69.61
Glass	70.37	70.37	68.52	70.37	**77.78**	61.09	58.59	56.89	58.69	**65.64**
Heart-statlog	76.47	**91.18**	88.24	82.35	80.88	76.45	**90.89**	88.19	82.29	80.68
Hypothyroid	99.36	99.47	**99.58**	99.47	**99.58**	71.28	72.34	**73.15**	72.03	72.85
Ionosphere	88.64	90.91	92.05	86.36	**93.18**	87.36	90.18	90.92	84.57	**92.29**
Iris	**97.37**	89.47	84.21	92.11	89.47	**97.33**	89.18	83.77	91.95	89.18
Letter	87.04	89.82	93.80	91.68	**94.84**	87.01	89.75	93.77	91.66	**94.82**
Liver-Disorders	58.62	**83.91**	64.37	71.26	66.67	52.60	**83.10**	62.38	68.45	65.18
Monks-problem	40.74	**74.07**	29.63	20.37	33.33	40.66	**74.04**	29.41	20.12	32.76
Mushroom	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
Optdigits	89.68	91.81	96.01	93.52	**96.94**	89.69	91.85	96.02	93.53	**96.95**
Pendigits	95.49	97.63	98.36	97.38	**98.62**	95.49	97.65	98.37	97.40	**98.62**
Primary-tumor	42.35	**52.94**	37.65	44.71	42.35	16.70	**21.36**	15.06	18.26	16.70
Segment	95.85	97.06	96.89	96.19	**98.27**	95.87	97.05	96.88	96.20	**98.27**
Sick	97.88	**99.15**	98.62	98.73	98.83	91.62	**96.14**	94.17	94.40	95.14
Sonar	76.92	76.92	**84.62**	76.92	73.08	76.36	76.89	**84.24**	76.36	72.04
Soybean	91.23	90.64	92.40	91.23	**93.57**	89.69	91.31	93.21	**95.02**	94.08
Vehicle	75.94	**77.83**	71.23	76.89	74.06	75.11	**77.86**	70.89	76.57	73.77
Vowel	74.60	80.65	**85.89**	82.26	83.47	73.12	80.86	**85.83**	82.17	83.31

Table 3
Accuracy and *f*-score results obtained by various ensembles of Naïve Bayes classifier over test data.

	Accuracy					*f*-Score				
Dataset	**Orig**	**Alloc**	**MB**	**Bag**	**Ada**	**Orig**	**Alloc**	**MB**	**Bag**	**Ada**
Anneal	73.78	**89.33**	77.33	76.00	73.78	51.37	**67.85**	57.75	53.98	51.37
Audiology	71.93	63.16	**77.19**	70.18	**77.19**	23.53	20.04	35.27	23.24	**36.94**
Autos	51.92	**55.77**	51.92	**55.77**	53.85	50.38	**54.44**	50.18	51.88	51.47
Balance-scale	89.17	**94.90**	89.81	89.17	89.81	62.02	**91.20**	62.46	62.02	62.46
Breast-cancer	72.22	**76.39**	70.83	72.22	70.83	65.38	**70.11**	63.08	65.38	63.08
Breast-w	95.43	96.00	95.43	95.43	**96.57**	94.97	95.61	94.97	94.97	**96.20**
Car	84.26	87.27	87.27	85.42	**91.20**	57.78	78.75	68.14	62.63	**80.87**
Colic	75.00	**81.52**	78.26	75.00	79.35	73.63	**79.16**	76.67	73.63	77.97
Credit-a	79.77	81.50	**82.08**	81.50	81.50	78.74	80.89	**81.36**	80.80	81.11
Diabetes	78.13	**93.23**	76.56	77.08	76.56	74.55	**92.62**	72.85	73.33	71.79
Ecoli	79.76	80.95	80.95	**82.14**	79.76	59.08	56.97	60.24	**61.43**	59.08
Eye_movements	44.26	**59.25**	44.26	44.15	44.26	42.69	**59.93**	42.69	42.55	42.69
Glass	50.00	50.00	51.85	42.59	**55.56**	40.75	34.50	42.17	34.01	**43.40**
Heart-Statlog	86.76	**89.71**	86.76	85.29	86.76	86.62	**89.60**	86.62	85.18	86.62
Hypothyroid	95.23	**95.65**	95.55	95.33	95.23	55.80	**58.67**	57.69	56.38	55.80
Ionosphere	80.68	79.55	85.23	82.95	**94.32**	79.74	78.65	83.93	81.93	**93.82**
Iris	92.11	89.47	92.11	94.74	**97.37**	92.00	89.18	92.00	94.66	**97.33**
Letter	63.56	**74.26**	63.72	63.58	63.56	63.07	**74.28**	63.30	63.05	63.07
Liver-Disorders	62.07	**82.76**	58.62	66.67	65.52	62.05	**82.30**	50.63	66.51	64.46
Monks-problem	38.89	**71.30**	39.81	41.67	38.89	38.55	**71.27**	39.69	41.06	38.55
Mushroom	94.83	95.42	99.66	94.88	**100.00**	94.80	95.40	99.65	94.85	**100.00**
Optdigits	90.18	**93.95**	90.82	90.04	90.18	90.24	**93.98**	90.88	90.10	90.24
Pendigits	85.01	**90.50**	84.64	84.79	85.01	84.56	**90.27**	84.08	84.33	84.56
Primary-tumor	48.24	**61.18**	48.24	48.24	48.24	20.25	**31.73**	20.25	20.42	20.25
Segment	81.83	**89.62**	81.83	81.83	81.83	79.33	**89.61**	79.33	79.37	79.33
Sick	92.79	**97.99**	93.00	92.68	92.26	76.24	**90.59**	76.39	75.75	74.34
Sonar	69.23	80.77	**82.69**	69.23	**82.69**	68.48	80.74	**82.63**	68.82	**82.63**
Soybean	91.81	**93.57**	92.40	92.40	91.23	95.16	**95.30**	94.75	95.07	93.94
Vehicle	47.64	**53.30**	47.64	49.06	47.64	44.70	**51.01**	44.70	45.40	44.70
Vowel	62.10	66.94	68.15	60.89	**75.81**	62.29	67.58	68.15	60.93	**75.44**

Table 4
Accuracy and *f*-score results obtained by various ensembles of IBk classifier over test data.

	Accuracy					*f*-Score				
Dataset	**Orig**	**Alloc**	**MB**	**Bag**	**Ada**	**Orig**	**Alloc**	**MB**	**Bag**	**Ada**
Anneal	96.00	**96.89**	96.00	95.11	96.00	78.64	**80.51**	78.64	77.65	78.64
Audiology	73.68	70.18	73.68	**75.44**	73.68	42.95	30.32	42.95	**43.61**	42.95
Autos	**75.00**	71.15	**75.00**	**75.00**	**75.00**	**66.76**	63.12	**66.76**	**66.76**	**66.76**
Balance-scale	84.71	**92.36**	84.71	85.35	84.71	59.72	**88.40**	59.72	59.93	59.72
Breast-cancer	73.61	73.61	72.22	**75.00**	70.83	64.14	64.14	62.96	**66.67**	64.20
Breast-w	94.29	**97.14**	94.29	94.29	94.29	93.55	**96.79**	93.55	93.55	93.55
Car	91.90	**92.36**	91.90	**92.36**	91.90	78.02	**78.50**	78.02	77.30	78.02
Colic	75.00	**79.35**	75.00	75.00	75.00	72.64	**77.07**	72.64	72.64	72.64
Credit-a	82.08	83.82	82.08	**85.55**	82.08	81.73	83.35	81.73	**85.27**	81.73
Diabetes	70.31	**92.71**	70.31	72.40	70.31	66.73	**91.86**	66.73	68.82	66.73
Ecoli	78.57	**79.76**	78.57	**79.76**	78.57	52.29	**55.73**	52.29	52.85	52.29
Eye_movements	60.53	**70.12**	60.53	60.24	60.53	61.17	**72.22**	61.17	60.86	61.17
Glass	64.81	**68.52**	64.81	64.81	64.81	53.98	**57.13**	53.98	52.45	53.98
Heart-statlog	79.41	**89.71**	79.41	79.41	79.41	79.39	**89.60**	79.39	79.39	79.39
Hypothyroid	90.99	**91.83**	90.99	91.41	90.99	43.23	**48.87**	43.23	43.51	43.23
Ionosphere	80.68	81.82	80.68	**84.09**	80.68	75.96	77.65	75.96	**80.44**	75.96
Iris	**92.11**	89.47	**92.11**	**92.11**	**92.11**	**91.95**	89.18	**91.95**	**91.95**	**91.95**
Letter	95.64	**97.18**	95.64	95.64	95.64	95.62	**97.15**	95.62	95.62	95.62
Liver-Disorders	70.11	**87.36**	70.11	67.82	70.11	69.20	**86.91**	69.20	67.08	69.20
Monks-problem	12.04	**53.70**	12.04	12.04	12.04	12.03	**53.64**	11.85	11.97	11.42
Mushroom	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
Optdigits	98.36	**99.07**	98.36	98.58	98.36	98.37	**99.08**	98.37	98.58	98.37
Pendigits	99.16	**99.49**	99.16	99.16	99.16	99.16	**99.49**	99.16	99.16	99.16
Primary-tumor	41.18	**56.47**	42.35	38.82	38.82	23.08	**31.89**	23.32	21.85	23.04
Segment	96.19	**97.23**	96.19	96.02	96.19	96.18	**97.22**	96.18	96.01	96.18
Sick	96.61	**98.30**	96.61	95.76	96.61	84.29	**92.00**	84.29	79.65	84.29
Sonar	73.08	**84.62**	73.08	73.08	73.08	72.92	**84.41**	72.92	72.92	72.92
Soybean	90.06	**91.23**	90.64	90.64	90.06	94.43	94.39	**94.64**	**94.64**	94.43
Vehicle	69.81	**75.47**	69.81	69.34	69.81	70.06	**75.52**	70.06	69.58	70.06
Vowel	**96.37**	**96.37**	**96.37**	**96.37**	**96.37**	96.26	**96.29**	96.26	96.26	96.26

Table 5
Accuracy and *f*-score results obtained by various ensembles of SMO classifier over test data.

	Accuracy					*f*-Score				
Dataset	**Orig**	**Alloc**	**MB**	**Bag**	**Ada**	**Orig**	**Alloc**	**MB**	**Bag**	**Ada**
Anneal	86.67	**89.78**	88.44	87.56	87.11	70.01	**71.97**	71.24	70.66	69.67
Audiology	**84.21**	75.44	80.70	80.70	**84.21**	42.12	31.73	37.37	36.88	**46.09**
Autos	69.23	**71.15**	**71.15**	69.23	63.46	47.12	50.48	48.93	47.51	**57.76**
Balance-scale	87.90	**96.82**	88.54	89.81	87.90	61.13	**92.57**	61.57	62.47	61.13
Breast-cancer	68.06	66.67	**70.83**	**70.83**	69.44	58.17	58.46	60.37	**61.81**	60.68
Breast-w	**97.14**	**97.14**	**97.14**	96.57	**97.14**	96.87	96.84	**96.87**	96.23	**96.87**
Car	93.29	93.29	**94.21**	91.90	93.06	88.39	87.02	**91.10**	84.53	86.83
Colic	79.35	78.26	76.09	**81.52**	76.09	77.40	75.27	73.24	**79.48**	73.65
Credit-a	83.82	**86.13**	82.66	84.39	78.61	83.80	**86.02**	82.49	84.37	78.04
Diabetes	79.69	**90.10**	80.21	80.21	79.69	75.55	**88.21**	76.05	75.79	75.55
Ecoli	79.76	78.57	**85.71**	79.76	**85.71**	47.14	46.22	**56.31**	47.14	53.65
Eye_movements	50.91	**62.07**	50.91	51.17	50.91	51.49	**64.58**	51.49	51.67	51.49
Glass	59.26	**64.81**	50.00	61.11	55.56	35.15	**49.32**	29.95	35.50	33.35
Heart-statlog	86.76	**92.65**	86.76	86.76	86.76	86.69	**92.52**	86.69	86.69	86.69
Hypothyroid	93.74	94.27	93.64	93.74	**94.80**	41.75	45.68	41.27	41.75	**51.53**
Ionosphere	88.64	88.64	90.91	90.91	**94.32**	86.64	87.14	89.52	89.52	**93.63**
Iris	92.11	84.21	**94.74**	86.84	92.11	91.95	83.33	**94.66**	86.32	91.95
Letter	81.94	**87.02**	81.88	82.04	81.94	81.96	**87.06**	81.87	82.10	81.96
Liver-Disorders	57.47	**86.21**	58.62	58.62	70.11	36.50	**84.97**	43.51	39.40	67.43
Monks-problem	42.59	**60.19**	43.52	43.52	40.74	42.59	**60.18**	42.41	41.30	38.18
Mushroom	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
Optdigits	97.86	**98.43**	98.01	97.94	97.94	97.87	**98.44**	98.01	97.94	97.94
Pendigits	97.71	**98.73**	98.00	97.89	98.00	97.70	**98.72**	97.99	97.88	97.97
Primary-tumor	47.06	**56.47**	43.53	43.53	47.06	21.40	**30.08**	20.16	19.82	21.40
Segment	92.04	**94.64**	91.87	92.39	92.04	91.97	**94.66**	91.80	92.28	91.97
Sick	93.96	**98.41**	93.96	93.85	94.91	48.44	**92.00**	48.44	48.41	77.95
Sonar	76.92	75.00	76.92	**80.77**	76.92	76.04	74.99	76.04	**80.51**	75.63
Soybean	92.40	91.23	**93.57**	92.98	92.98	95.70	94.84	**96.17**	95.95	95.93
Vehicle	73.11	**77.36**	72.17	72.64	73.11	72.13	**75.54**	71.34	71.59	72.13
Vowel	66.13	70.97	70.16	64.92	**77.02**	65.79	71.30	70.42	64.81	**77.21**

Table 6
Accuracy and *f*-score results obtained by various ensembles of OneR classifier over test data.

	Accuracy					*f*-Score				
Dataset	**Orig**	**Alloc**	**MB**	**Bag**	**Ada**	**Orig**	**Alloc**	**MB**	**Bag**	**Ada**
Anneal	83.56	**85.78**	83.56	83.56	83.56	31.71	**39.77**	31.71	31.71	36.46
Audiology	47.37	**52.63**	47.37	47.37	47.37	5.46	**10.86**	5.46	5.46	5.46
Autos	63.46	51.92	**67.31**	61.54	65.38	42.86	37.71	**48.40**	39.35	46.60
Balance-scale	58.60	64.33	**77.71**	74.52	73.89	40.53	**69.05**	54.04	51.79	51.35
Breast-cancer	72.22	68.06	**73.61**	70.83	72.22	62.96	59.56	58.98	65.19	**66.39**
Breast-w	89.14	96.57	92.57	92.57	**93.71**	88.26	**96.13**	91.72	91.65	93.00
Car	70.14	**73.84**	70.14	70.14	70.14	20.61	**47.54**	20.61	20.61	20.61
Colic	75.00	**77.17**	75.00	75.00	73.91	**73.89**	72.17	**73.89**	**73.89**	70.32
Credit-a	**87.86**	**87.86**	**87.86**	**87.86**	82.66	**87.85**	**87.85**	**87.85**	**87.85**	82.61
Diabetes	71.35	**91.15**	71.88	74.48	72.92	63.95	**89.83**	65.22	67.09	69.29
Ecoli	64.29	**67.86**	64.29	66.67	65.48	27.36	**43.53**	26.94	28.62	25.95
Eye_movements	53.66	**69.75**	58.27	55.30	63.20	52.68	**71.57**	57.81	54.56	62.90
Glass	53.70	**57.41**	50.00	53.70	50.00	31.00	**44.69**	24.41	26.22	24.22
Heart-statlog	76.47	**89.71**	77.94	76.47	82.35	76.39	**89.60**	77.35	76.39	82.29
Hypothyroid	96.92	**97.77**	97.03	96.92	96.71	60.97	**67.55**	62.35	61.60	62.85
Ionosphere	82.95	79.55	**86.36**	81.82	85.23	79.75	75.96	**83.62**	77.08	82.45
Iris	92.11	**97.37**	94.74	94.74	89.47	92.00	**97.33**	94.66	94.66	89.18
Letter	17.50	**29.36**	17.34	17.50	17.34	8.80	**21.12**	8.69	8.80	8.69
Liver-disorders	58.62	**87.36**	56.32	63.22	59.77	57.35	**86.31**	53.62	60.45	58.36
Monks-problem	40.74	**74.07**	39.81	43.52	42.59	40.66	**74.04**	37.89	38.45	40.95
Mushroom	98.82	98.82	98.82	98.82	**99.90**	98.82	98.82	98.82	98.82	**99.90**
Optdigits	25.98	**40.28**	25.55	26.98	25.55	20.20	**37.45**	19.55	20.95	19.55
Pendigits	39.67	**64.48**	38.21	42.83	38.21	35.61	**63.14**	33.32	39.00	33.32
Primary-tumor	27.06	**42.35**	24.71	27.06	24.71	3.79	**9.98**	3.82	3.69	3.82
Segment	63.15	**82.70**	74.39	65.22	79.76	63.12	**83.29**	73.47	64.97	79.55
Sick	96.18	**98.41**	95.55	96.18	95.86	85.54	**92.00**	81.60	85.54	81.64
Sonar	69.23	**75.00**	67.31	73.08	55.77	68.48	**74.54**	65.13	72.71	54.95
Soybean	40.94	**57.89**	41.52	41.52	41.52	14.92	**29.37**	15.78	15.78	15.78
Vehicle	51.42	**58.96**	50.47	53.30	56.13	50.48	**58.79**	49.49	52.72	55.67
Vowel	33.47	**50.81**	29.44	37.50	29.44	33.33	**50.63**	28.95	36.46	28.95

Table 7
Accuracy and *f*-score results obtained by various ensembles of NBTree classifier over test data.

	Accuracy					*f*-Score				
Dataset	**Orig**	**Alloc**	**MB**	**Bag**	**Ada**	**Orig**	**Alloc**	**MB**	**Bag**	**Ada**
Anneal	94.67	96.00	**98.22**	96.89	97.78	77.98	79.13	**81.28**	80.03	80.83
Audiology	73.68	70.18	78.95	77.19	**84.21**	26.74	27.99	39.97	36.65	**45.64**
Autos	76.92	63.46	**80.77**	73.08	76.92	55.67	56.00	**71.60**	67.52	69.01
Balance-scale	78.34	85.99	84.08	**87.26**	81.53	54.47	**84.79**	58.45	60.68	56.87
Breast-cancer	72.22	69.44	70.83	**75.00**	66.67	61.50	57.65	61.81	**65.35**	53.80
Breast-w	96.57	**99.43**	96.57	96.57	96.57	96.23	**99.36**	96.23	96.23	96.23
Car	90.74	90.74	**94.44**	92.13	93.52	79.84	74.67	**85.09**	79.28	81.59
Colic	75.00	79.35	79.35	**81.52**	79.35	72.24	75.85	76.30	**79.16**	77.07
Credit-a	**86.71**	86.13	**86.71**	85.55	81.50	86.44	86.05	**86.49**	85.31	81.36
Diabetes	79.17	**93.75**	80.73	80.73	77.60	76.18	**92.97**	77.87	77.47	74.06
Ecoli	83.33	83.33	**86.90**	82.14	79.76	50.15	54.35	**62.07**	47.22	58.61
Eye_movements	63.86	73.74	63.86	**74.76**	63.86	64.70	75.49	64.70	**75.52**	64.70
Glass	**72.22**	70.37	68.52	66.67	70.37	**61.68**	53.78	55.22	52.30	54.86
Heart-statlog	86.76	86.76	85.29	85.29	**88.24**	86.53	86.62	85.09	85.18	**88.14**
Hypothyroid	99.26	99.47	**99.68**	99.47	99.58	72.31	73.13	**73.70**	72.59	73.68
Ionosphere	88.64	82.95	**93.18**	92.05	90.91	87.36	81.46	**92.29**	90.92	89.52
Iris	89.47	**92.11**	89.47	89.47	89.47	89.18	**91.95**	89.18	89.18	89.18
Letter	86.02	89.32	94.30	94.00	**94.72**	86.04	89.26	94.28	94.01	**94.70**
Liver-Disorders	63.22	**85.06**	68.97	73.56	65.52	58.62	**84.22**	66.41	71.39	62.93
Monks-problem	25.00	**68.52**	25.93	18.52	16.67	24.94	**68.51**	25.70	18.52	16.55
Mushroom	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
Optdigits	88.75	90.60	96.80	**97.15**	**97.15**	88.73	90.62	96.80	**97.16**	**97.16**
Pendigits	95.31	95.56	98.29	98.22	**98.62**	95.30	95.55	98.30	98.23	**98.62**
Primary-tumor	45.88	**54.12**	43.53	47.06	45.88	17.75	22.49	18.72	18.40	**23.17**
Segment	94.98	95.85	98.10	97.92	**98.27**	94.99	95.82	98.09	97.92	**98.27**
Sick	97.14	98.20	98.62	98.30	**98.94**	87.50	91.43	93.98	92.41	**95.48**
Sonar	84.62	75.00	78.85	**90.38**	88.46	84.41	74.99	78.46	**90.09**	88.02
Soybean	**93.57**	91.81	**93.57**	**93.57**	92.40	94.31	94.51	**96.21**	95.69	95.60
Vehicle	67.92	**74.53**	71.23	71.23	72.64	68.19	**74.10**	70.21	70.23	71.96
Vowel	87.90	**91.13**	87.90	**91.13**	87.90	87.71	**91.34**	87.71	91.03	87.71

Table 8
Average accuracy results obtained by various ensembles of different classifiers over test data.

Method →	Original (none)		Allocation		MultiBoost		Bagging		AdaBoost	
Classifier ↓	Avg	Rank	Avg	Rank	Avg	Rank	Avg	Rank	Avg	Rank
J48	80.95	14.99	**84.74**	**11.26**	82.10	16.82	82.17	16.50	82.78	15.90
Naïve Bayes	74.29	16.99	**80.51**	**14.30**	75.82	17.31	74.70	17.13	76.89	17.37
IBk	80.08	18.90	**85.24**	**12.89**	80.09	18.85	80.35	19.10	79.91	19.11
SMO	79.99	15.94	**83.49**	**12.91**	80.16	16.64	80.14	15.91	80.66	16.03
OneR	63.39	22.59	**72.31**	**19.09**	64.50	23.65	65.01	22.40	64.49	23.26
NBTree	81.26	16.40	**84.43**	**12.03**	83.12	17.01	83.56	17.24	82.50	18.09
All classifiers	76.66	18.69	**81.79**	**14.51**	77.63	19.35	77.65	19.00	77.87	19.27

Table 9
Average *f*-score results obtained by various ensembles of different classifiers over test data.

Method →	Original (none)		Allocation		MultiBoost		Bagging		AdaBoost	
Classifier ↓	Avg	Rank	Avg	Rank	Avg	Rank	Avg	Rank	Avg	Rank
J48	72.61	19.92	**78.14**	**18.71**	74.98	20.75	74.65	20.65	75.88	20.06
Naïve Bayes	65.62	20.62	**73.74**	**20.26**	67.42	20.39	66.12	20.79	68.78	20.99
IBk	72.48	22.34	**78.42**	**19.67**	72.45	22.36	72.57	22.43	72.46	22.40
SMO	68.98	22.92	**75.67**	**20.55**	69.58	23.11	69.01	23.01	72.14	20.98
OneR	50.64	27.68	**62.67**	**25.88**	51.17	28.45	51.73	28.08	51.77	28.38
NBTree	72.92	21.83	**77.47**	**19.40**	76.07	21.07	75.85	22.33	75.51	21.69
All classifiers	67.21	23.72	**74.35**	**21.30**	68.61	24.04	68.32	24.11	69.42	23.75

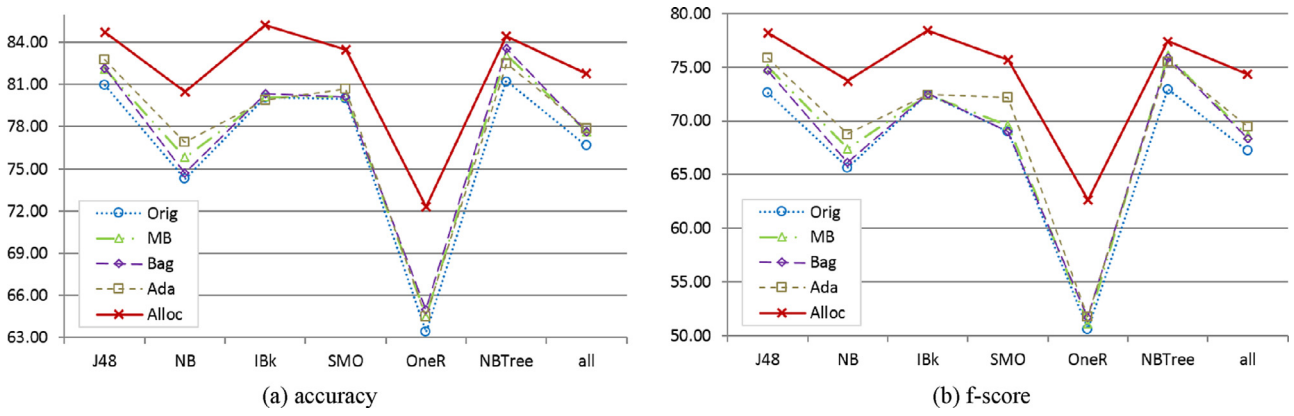


Fig. 5. Accuracy and *f*-score results (averaged over all 30 data sets) of five methods (on six single classifiers and in total).

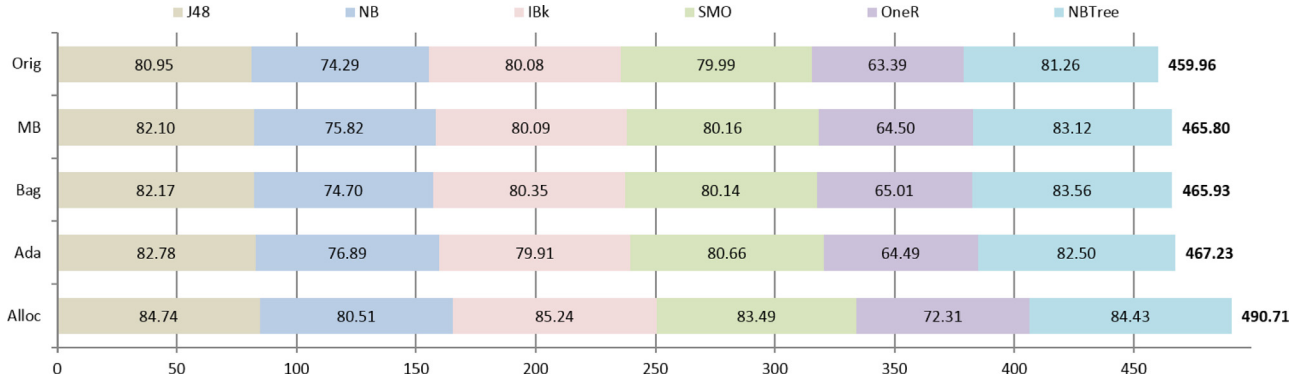


Fig. 6. Aggregated average accuracy (for all 30 data sets) from all six classifiers.

seen that our allocation method performed the best on average. Especially, there are some data sets (diabetes, letter, liver-disorders, monks-problem, and primary-tumor), where our allocation method outperforms all the other methods considerably. It can be also seen that there are some differences between the classification algorithms used. While the worst results were obtained when using OneR classifier, the use of J48, IBk and NBTree provided the most accurate results. The use of Naive Bayes and SMO classifiers provided accuracies which were somewhere in the middle.

Finally, we performed the statistical tests to see whether the differences between the methods are statistically significant. For this purpose, we compared the accuracy and *f*-score results of all five methods (original classification results with no ensemble, our allocation method, MultiBoost, bagging and AdaBoost) separately

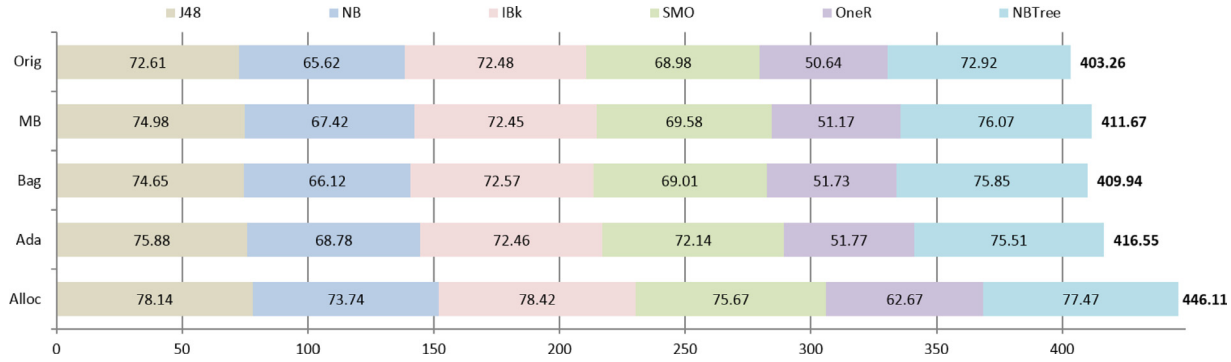
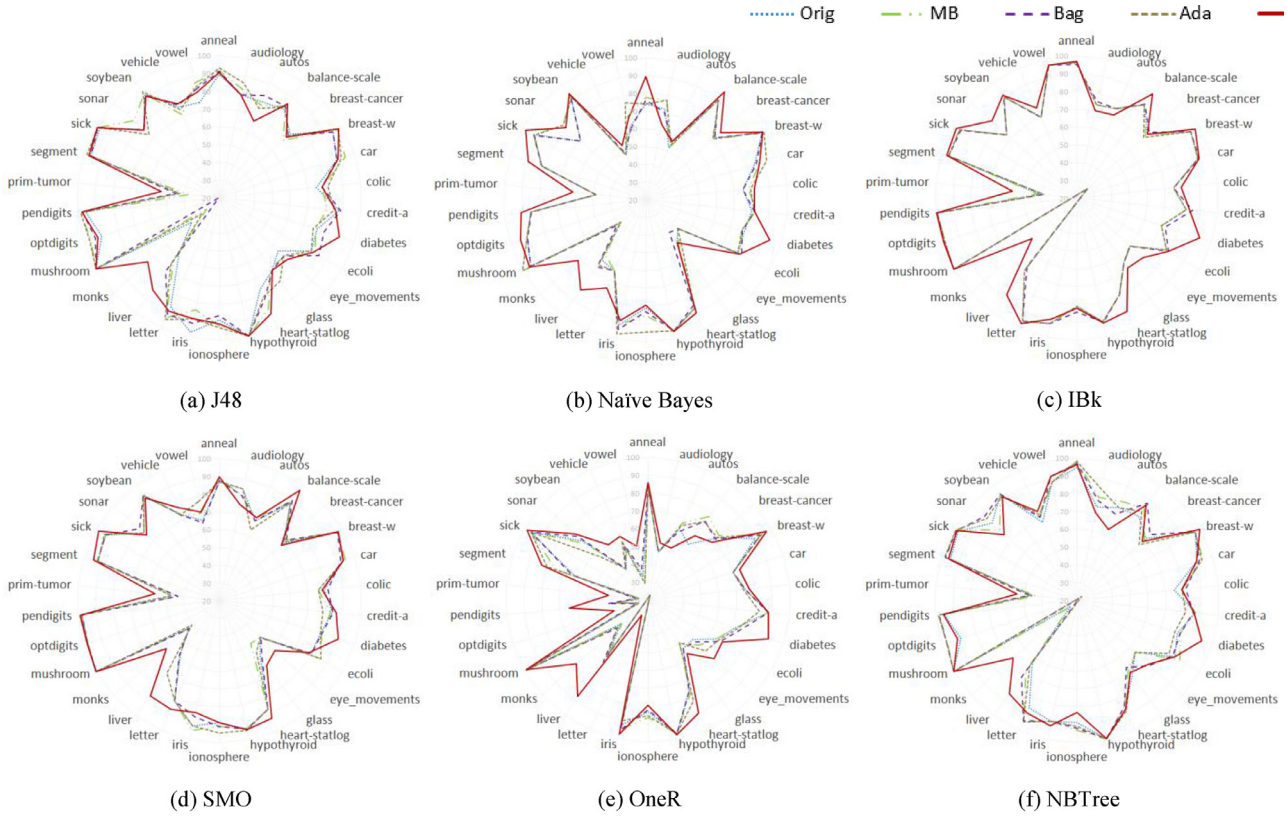


Fig. 7. Aggregated average *f*-score (for all 30 data sets) from all six classifiers.



(a) J48 (b) Naïve Bayes (c) IBk

(d) SMO (e) OneR (f) NBTree

Fig. 8. Comparison of accuracy results obtained by five methods on different classifiers over test data.

for all six classifiers and also for the combined results of all classifiers together.

We first used the Friedman test [12] to compare the differences between accuracies of all five methods as is recommended by Demšar [7]. As we can see from Table 10, there are significant differences between the five methods for all six single classifiers and also in the case of all classifiers combined. As our allocation method achieved the best average accuracy results for each of the single classifiers, we next performed the Wilcoxon signed ranks test for comparing our allocation method with each of the four remaining ones. To counteract the problem of multiple comparisons the Holm-Bonferroni correction [19] was applied. The results are presented in Table 10 (all the results, where the advantage of our allocation method is statistically significant, are remarked in bold and marked with an asterisk). In the case of IBk and OneR, as well as when all classifiers are considered, our allocation method achieved significantly better accuracy than all the other methods. In the case of Naive Bayes, the accuracy is significantly better than all but AdaBoost. In the case of J48 and SMO our method significantly outperformed the original results, while in the case of NBTree no statistically significant differences were found.

Similarly, the Friedman test confirmed the statistically significant differences between *f*-score of all five methods (Table 11) for all the classifiers. The results of the Wilcoxon signed ranks test (Table 11) show, that our allocation method achieved significantly better *f*-score than all other methods when using Naive Bayes, IBk and OneR classifiers, as well as when all the classifiers together are considered. In the case of SMO, only AdaBoost was not significantly worse than our method. In the case of J48, our allocation method significantly outperformed the original results, while in the case of NBTree no statistically significant differences were found.

3.3. Further analysis of the obtained results

Beside the basic comparison of accuracy and *f*-score results for all the five methods presented above, we performed several other

Table 10
Statistical test results obtained by comparing accuracy of five methods for different classifiers.

Classifier	Friedman test all 5 compared	Wilcoxon signed ranks test with Holm–Bonferroni correction Alloc vs. Orig	Alloc vs. MB	Alloc vs. Bag	Alloc vs. Ada
J48	**0.005***	**0.007***	0.419	0.290	0.936
Naive Bayes	**0.000***	**0.000***	**0.006***	**0.000***	0.052
IBk	**0.000***	**0.001***	**0.000***	**0.006***	**0.000***
SMO	**0.016***	**0.007***	0.037	0.020	0.036
OneR	**0.000***	**0.000***	**0.001***	**0.000***	**0.001***
NBTree	**0.024***	0.080	0.909	0.631	0.605
All classifiers	**0.000***	**0.000***	**0.000***	**0.000***	**0.000***

Statistically significant differences (*$p<0.05$) after Holm–Bonferroni correction are outlined.

Table 11
Statistical test results obtained by comparing *f*-score of five methods for different classifiers.

Classifier	Friedman test all 5 compared	Wilcoxon signed ranks test with Holm–Bonferroni correction Alloc vs. Orig	Alloc vs. MB	Alloc vs. Bag	Alloc vs. Ada
J48	**0.001***	**0.004***	0.239	0.133	0.802
Naive Bayes	**0.000***	**0.000***	**0.006***	**0.000***	**0.037***
IBk	**0.000***	**0.001***	**0.001***	**0.003***	**0.001***
SMO	**0.029***	**0.003***	**0.019***	**0.008***	0.110
OneR	**0.000***	**0.000***	**0.000***	**0.000***	**0.000***
NBTree	**0.000***	0.020	0.770	0.891	0.804
All classifiers	**0.000***	**0.000***	**0.000***	**0.000***	**0.000***

Statistically significant differences (*$p<0.05$) after Holm–Bonferroni correction are outlined.

analyses of the obtained results, which should provide additional insights about our allocation method. Supported by very good results of the basic comparison, which indicated the power of our allocation method, we wanted to check whether there are some situations – be the characteristics of a single dataset, the chosen basic classification algorithm, or the results of other methods – where one could expect above average improvements when using our allocation method.

3.3.1. The correlation of accuracy and f-score between methods

First we checked how the accuracy of our allocation method correlates with the accuracy of the other approaches. For this purpose, we sorted all the 180 obtained accuracy results (30 data sets times six classifiers) according to the accuracy of the original classifiers (when no ensemble method is used) and plotted the accuracies of five methods compares (Fig. 9). It can be seen, that the improvements of our allocation method are larger towards the right side of the graph – where the accuracies are the lowest. These are also the situations where a method which improves the classification performance is the most needed.

To check whether this anticipation holds, we calculated the absolute and relative gains (the amount of improvement in accuracy over the original classifier) of each method for all 180 results:

$$absGain(i) = acc_x(i) - acc_{orig}(i)$$
$$relGain(i) = \frac{acc_x(i) - acc_{orig}(i)}{acc_{orig}(i)} \quad (11)$$

where x is the ensemble method and i is one of 180 accuracy results.

We split all 180 calculated gains to two groups: the first 90 into first group (the cases where the accuracies were better), and the last 90 into second group (the cases with the lower accuracies). The absolute accuracy improvements of our allocation method with regard to the original classifier are 1.1% on average for the first group (relative: 1.24) and 9.24% on average for the second group (relative: 23.61). Then we performed a nonparametric test – the Mann–Whitney U test [31], comparing the gains of the two groups. The results showed that there is a statistically significant difference ($p<0.001$) in both absolute and relative gains between the two groups. Similarly, we compared the gain of our allocation method also to the other three ensemble methods and the differences were statistically significant in all the cases. This means that our allocation method provides higher improvements of accuracy in the cases where classification accuracy of other methods is low.

Similarly, next we checked whether the same holds also for the *f*-score (Fig. 10). The statistical results of *f*-score gains, calculated in the same way as described above for the accuracy, revealed that there is a statistically significant difference ($p<0.001$) in both absolute and relative *f*-score gains between the two groups (high and

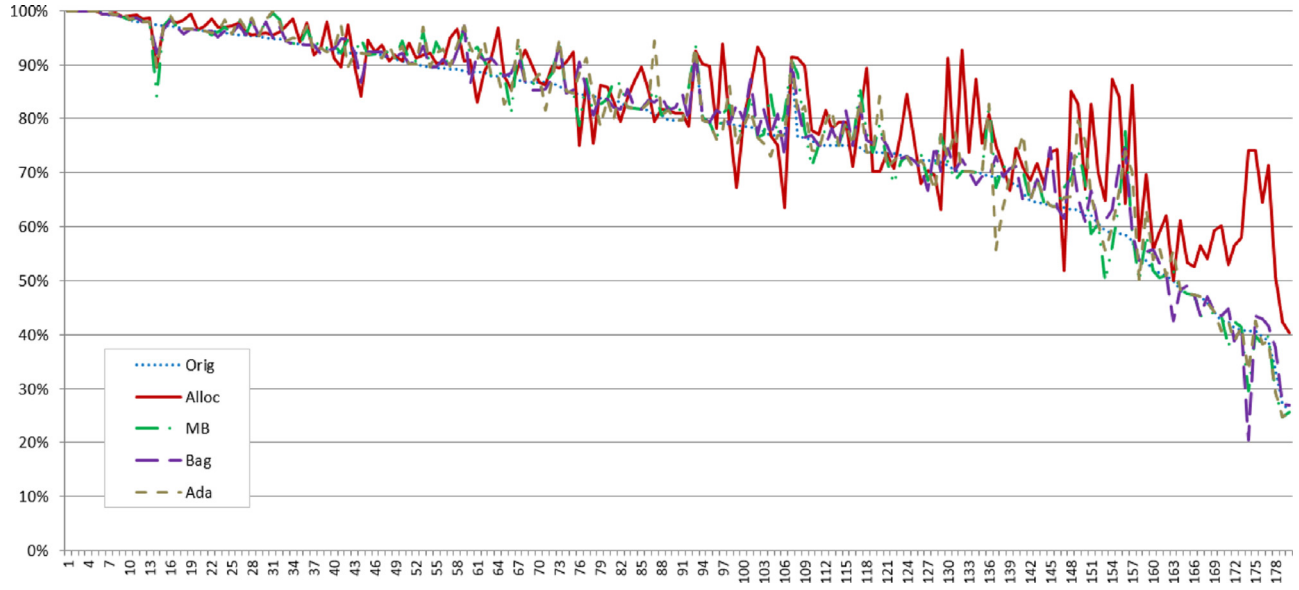


Fig. 9. Accuracy results obtained by five methods (all six classifiers over 30 data sets); sorted according to the accuracy of original classifiers (from highest to lowest).

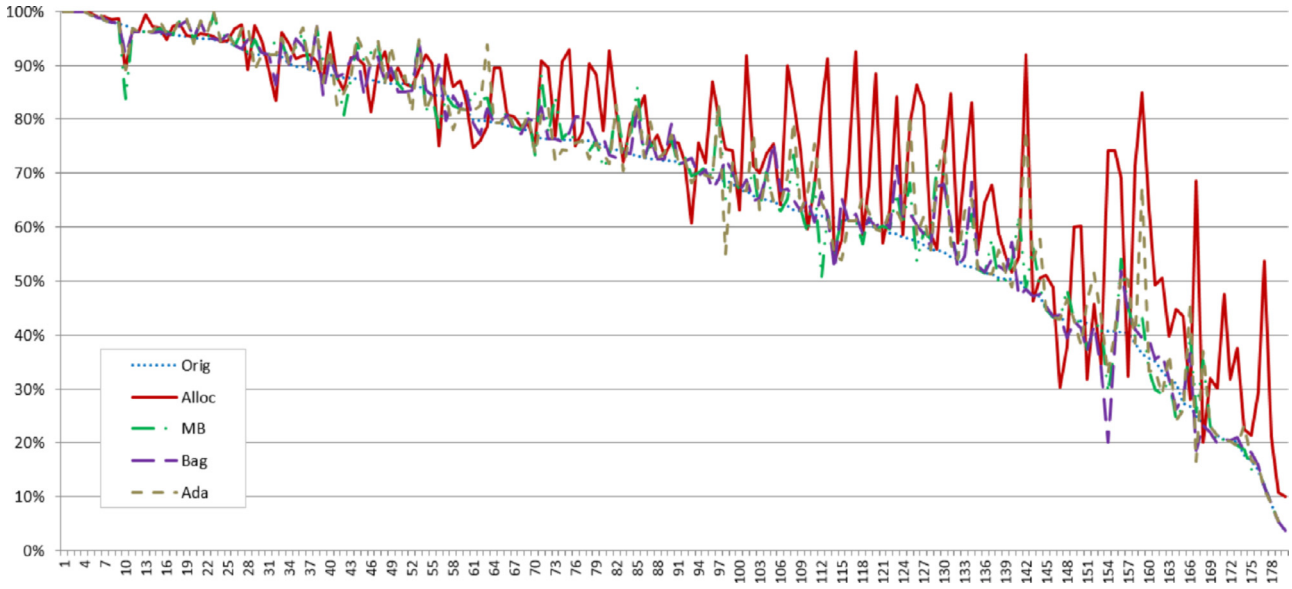


Fig. 10. *f*-Score results obtained by five methods (all six classifiers over 30 data sets); sorted according to the *f*-score of original classifiers (from highest to lowest).

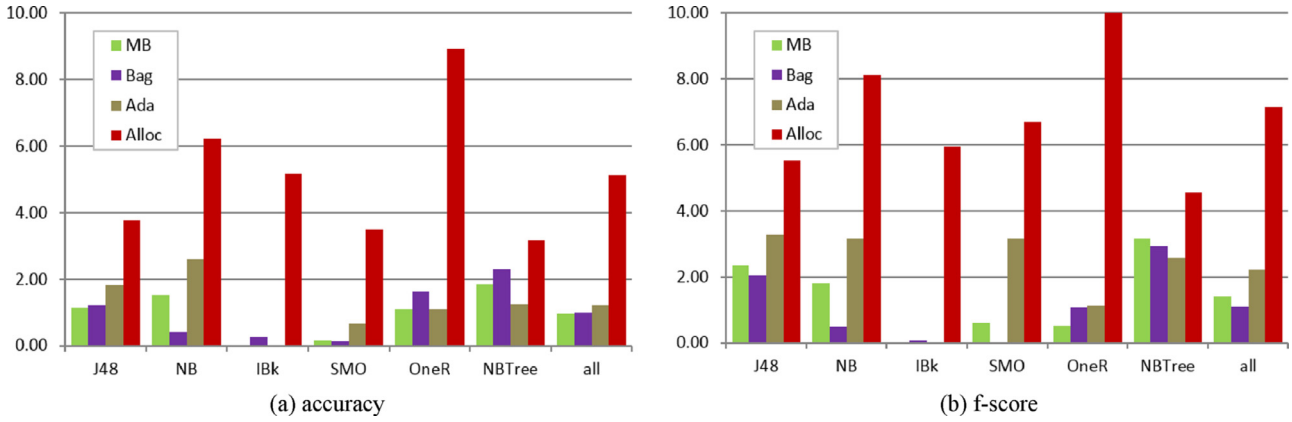


Fig. 11. Accuracy and *f*-score improvements (averaged over all 30 data sets) of five methods (on six single classifiers and in total).

low *f*-score results) of our allocation method with regard to all the other four approaches. Even more, both the absolute and relative improvements in *f*-score of our method were somewhat bigger than in the case of accuracy. Namely, the absolute *f*-score improvements of our allocation method with regard to the original classifiers are 1.15% on average for the first group (relative: 1.33) and 10.91% on average for the second group (relative: 31.8).

3.3.2. The improvements in accuracy and f-score for different classifiers

Next we checked how a basic classification algorithm influences the amount of improvement of accuracy and *f*-score in an ensemble method with regard to the original basic classifier. For this purpose, we calculated the absolute accuracy and *f*-score gains of the four ensemble methods with regard to the original classifiers of all 180 results (30 data sets times six classifiers) as described above and grouped them by basic classifiers. The average accuracy and *f*-score gains for each of the basic classifier, as well as the overall average gain, for all four ensemble methods are presented in Fig. 11.

It can be seen from Fig. 11 that our allocation method achieved the biggest improvements for all the basic classifiers, both in accuracy and *f*-score. It is interesting though that the improvements of our allocation method are the lowest in the case of both decision tree classifiers – NBTree and J48 – where the other three ensemble methods gain the most. On the other hand, our allocation method considerably improved the results of IBk and SMO classifiers, where all the other ensemble methods gain practically nothing.

3.3.2. The aggregated regret analysis

Next, we performed the aggregated regret analysis. Regret (also called opportunity loss) can be defined as the difference between an actual result (the accuracy result of one specific method in our case) and the result of the best possible method (a method with the best accuracy in our case):

$$regret_x(i) = \max(acc(i)) - acc_x(i) \tag{12}$$

where x is the used method, i is one of 180 accuracy results (30 data sets times six classifiers), and $max(acc(i))$ is the best accuracy of i-th result from all the methods.

The aggregated regret (at the point i) is then the sum of all the regrets for a single method:

$$aggregatedRegret_x(i) = \sum_{j=1}^{i} regret_x(j) \tag{13}$$

Similarly, the aggregated regret for the *f*-score can be calculated. Fig. 12 illustrates both the calculated aggregated regrets for

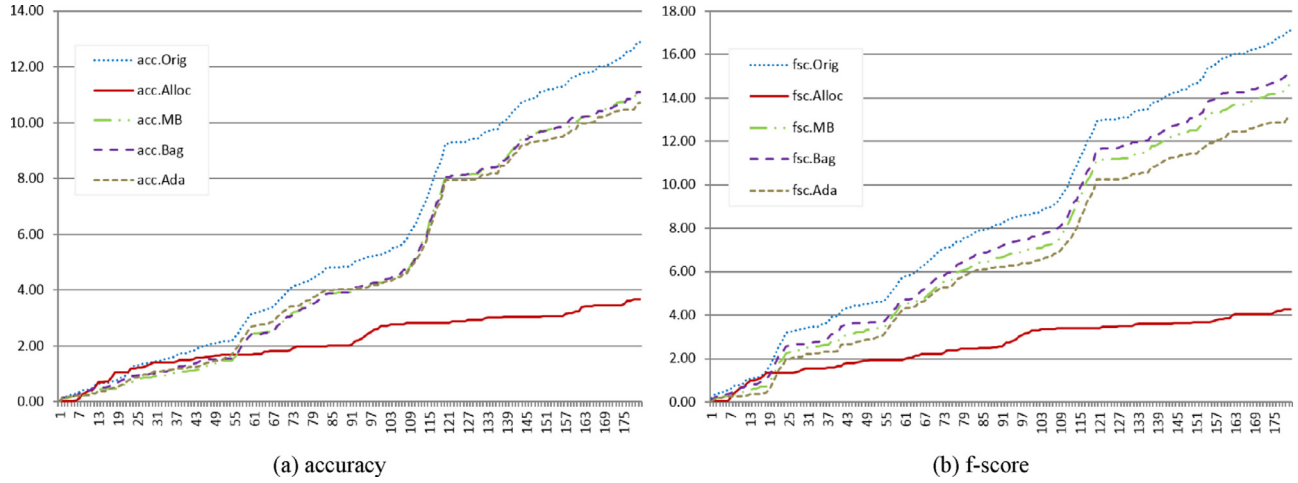


Fig. 12. Aggregated regret of five methods (for 30 datasets times six basic classifiers).

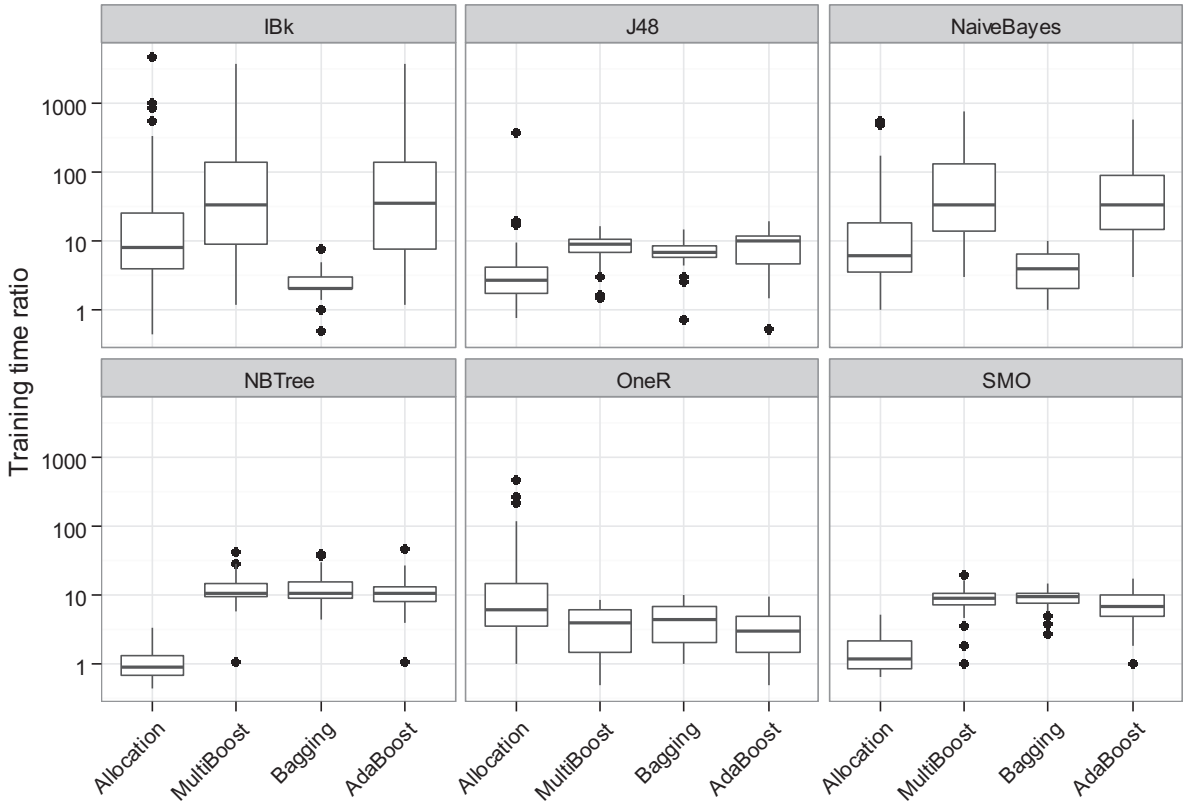


Fig. 13. Comparison of training times of the ensemble methods in combination with each of the six classification methods (for 30 datasets) on a logarithmic scale.

accuracy and *f*-score for all five methods. It can be seen that both the aggregated accuracy regret and the aggregated *f*-score regret of our allocation method are much lower than the aggregated regrets of the other methods. As expected, the aggregated regrets of the original classifiers are the highest both for accuracy and *f*-score. The aggregated accuracy regrets of the three existing ensemble methods are almost identical. On the other hand, the aggregated *f*-score results reveal bigger differences also between these three. As it turned out, the lowest aggregated *f*-score regret (which is the best) of the three existing ensemble methods was achieved with AdaBoost followed by MultiBoost and the highest aggregated *f*-score results were achieved with bagging – which is correlated with the results of aggregated average *f*-score s (see Fig. 7).

3.4. The analysis of training time

Lastly, we analyzed the training time cost of allocation method and compared this to the training time costs of other ensemble methods. Fig. 13 shows the ratios of computation times for training each ensemble method on logarithmic scale. Times presented are calculated as ratios of ensemble method training time in comparison to original classifier computation time as is shown below in Eq. (14).

$$timeRatio_i = \frac{trainingTime_i}{trainingTime_{original}} \tag{14}$$

Table 12 presents statistical analysis of training time and results confirm the findings from the Fig. 13. Results show that allocation

Table 12
Statistical test results obtained by comparing training times of four methods for different classifiers.

Classifier	Friedman test all 5 compared	Wilcoxon signed ranks test with Holm–Bonferroni correction Alloc vs. MB	Alloc vs. Bag	Alloc vs. Ada
J48	**0.000***	**0.002***	**0.017***	**0.009***
Naive Bayes	**0.000***	**0.000***	0.000*	**0.001***
IBk	**0.000***	**0.001***	0.000*	**0.001***
SMO	**0.000***	**0.000***	**0.000***	**0.000***
OneR	**0.000***	0.002*	0.000*	0.000*
NBTree	**0.000***	**0.000***	**0.000***	**0.000***
All classifiers	**0.000***	**0.000***	**0.000***	**0.000***

Statistically significant differences ($*p<0.05$) after Holm–Bonferroni correction where allocation has lower training times are outlined.

method takes least time to train its classification model in combination with three out of six basic classifiers (J48, NBTree and SMO). In its original form, these three classification methods have the highest training times of all six used classifiers. This means that the allocation method has smallest time addition in comparison to other ensembles in combination with the most time consuming methods. The definition of allocation method is such that each classification instance is involved in training exactly two times – first in training of the anomaly detection model and second in training of the base classifiers used as micro-classifiers. In boosting methods (AdaBoost and MultiBoost) every instance is involved in training in every iteration (10 iterations in our experiment). Similarly, full set of instances is used in training on every bag (some instances are used in more bags and others are used less). Allocation training time is close to the training time of bagging model in combination with IBk (which is a lazy classifier with almost no training) and NaiveBayes, but it is still smaller than AdaBoost and MultiBoost. The last position in combination with OneR classifier can be explained by the significantly larger training time of anomaly detection model than training of the multiple OneR models (one model for each iteration or bag) and other ensembles have no overhead from training the anomaly detection model.

4. Conclusion

The use of meta-classification methods, like ensemble methods, improves the classification accuracy and other classification metrics in comparison to original classification algorithms without ensembles. This paper proposes a new ensemble method called allocation. The allocation method uses an allocator to split the dataset in multiple disjoint subsets and allocates them to one of the specialized micro classifiers. For this paper we used a one-class SVM anomaly detection algorithm for the allocator, which divided the training dataset into two disjoint subsets. For micro classifiers several existing classification algorithms were tested. The obtained results of the tests were compared to these existing classifiers without ensembles and to other ensemble methods. Aggregated results, grouped by classification method, show that our allocation method outperforms all other ensemble methods in overall accuracy and *f*-score, achieving the highest median and Friedman rank in both of the metrics, and for all the used original classification algorithms. Improvements were statistically significant for most of the classification algorithms in the accuracy comparison and even more for the *f*-score measurement. The lowest improvements, which were not statistically significant, have been achieved with the two decision tree classification algorithms (NBTree and J48). Considering that by definition ensemble methods, such as boosting, are designed for weak learners (such as classification rules or decision trees) [38], this fact is, of course, not surprising. Although the results do not show statistically significant differences with both decision tree classifiers, our allocation method achieved improvements also there. The strength of our method was further confirmed with the analysis of classification performance improvements among all ensemble methods, with the aggregated regret analysis, and with the training time analysis.

Experiments described in this paper were conducted with limited variation of the allocation method – only anomaly detection one-class SVM allocator was used to split the dataset into exactly two subsets, and the same classification algorithm was always used for both micro classifiers. Based on the obtained promising results, future research will focus towards using other allocation techniques, such as clustering, as suggested in [27]. In this way, the allocator may split the data into more than two subsets; one obvious option would be to use the number of micro classifiers equal to the number of classes. Another viable alternative could be the usage of different classification algorithms for each of the micro classifiers. After all, also a combination of classifiers in a form of any existing ensemble can be used to deal with the subset of data, split by the allocator.

References

[1] D. Aha, D. Kibler, Instance-based learning algorithms, Mach. Learn. 6 (1991) 37–66.
[2] K. Bache, M. Lichman, UCI Machine Learning Repository (2013).
[3] L. Breiman, Bagging predictors, Mach. Learn. 24 (1996) 123–140.
[4] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (27) (2011) 1–27:27.
[5] R.T. Clemen, Combining forecasts: a review and annotated bibliography, Int. J. Forecast. 5 (1989) 559–583.
[6] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (1995) 273–297.
[7] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[8] T.G. Dietterich, Ensemble methods in machine learning, in: Mult. Classif. Syst., Springer, 2000, pp. 1–15.
[9] C. Ferri, P. Flach, J. Hernández-Orallo, Delegating classifiers, in: Proceedings of the Twenty-First International Conference on Machine Learning - ICML '04, ACM Press, New York, New York, USA, 2004, p. 37.
[10] B. Frénay, M. Verleysen, Classification in the presence of label noise: a survey., IEEE Trans. Neural Netw. Learn. Syst. 25 (2014) 845–869.
[11] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1997) 119–139.
[12] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Am. Stat. Assoc. 32 (1937) 675–701.
[13] M. Galar, A. Fernández, E. Barrenechea, F. Herrera, EUSBoost: enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling, Pattern Recognit. 46 (2013) 3460–3471.
[14] N. García-Pedrajas, J. Maudes-Raedo, C. García-Osorio, J.J. Rodríguez-Díez, Supervised subspace projections for constructing ensembles of classifiers, Inf. Sci. 193 (2012) 1–21.
[15] J.V. Hansen, Combining predictors: comparison of five meta machine learning methods, Inf. Sci. 119 (1999) 91–105.
[16] L.K. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. Pattern Anal. Mach. Intell. 12 (1990) 993–1001.
[17] A. de Haro-García, N. García-Pedrajas, A divide-and-conquer recursive approach for scaling up instance selection algorithms, Data Min. Knowl. Discov. 18 (2008) 392–418.
[18] S. Hashem, optimal linear combinations of neural networks, Neural Networks. 10 (1997) 599–614.
[19] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. (1979) 65–70.
[20] R.C. Holte, Very simple classification rules perform well on most commonly used datasets, Mach. Learn. 11 (1993) 63–91.
[21] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptive mixtures of local experts, Neural Comput. 3 (1991) 79–87.
[22] G.H. John, P. Langley, Estimating continuous distributions in bayesian classifiers, in: Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, 1995, pp. 338–345.
[23] D.J.H. Jonathan, J. Oliver, On pruning and averaging decision trees, Proceedings of the Twelfth International Conference On Machine Learning (1995) 430–437.
[24] M.I. Jordan, R.A. Jacobs, Hierarchical mixtures of experts and the EM algorithm, Neural Comput. 6 (1994) 181–214.
[25] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, K.R.K. Murthy, Improvements to Platt's SMO Algorithm for SVM Classifier Design, Neural Comput. 13 (2001) 637–649.
[26] R. Kohavi, Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996, pp. 202–207.
[27] B. Krawczyk, M. Woźniak, B. Cyganek, Clustering-based ensembles for one-class classification, Inf. Sci. 264 (2014) 182–195.

[28] A. Krogh, J. Vedelsby, Neural network ensembles, cross validation, and active learning, Adv. Neural Inf. Process. Syst. (1995) 231–238.
[29] M. Lopes, F. Gouyon, A.L. Koerich, L.E.S. Oliveira, Selection of training instances for music genre classification, in: Proceedings of the 2010 20th International Conference on Pattern Recognition, IEEE, 2010, pp. 4569–4572.
[30] R. Maclin, D. Opitz, Popular ensemble methods: An empirical study. arXiv:1106.0257. (2011).
[31] H.B. Mann, D.R. Whitney, On a test of whether one of two random variables is stochastically larger than the other, Ann. Math. Stat. 18 (1947) 50–60.
[32] J.A. Olvera-López, J.A. Carrasco-Ochoa, J.F. Martínez-Trinidad, J. Kittler, A review of instance selection methods, Artif. Intell. Rev. 34 (2010) 133–143.
[33] D.W. Opitz, J.W. Shavlik, Generating accurate and diverse members of a neural-network ensemble, Adv. Neural Inf. Process. Syst. 8 (1996) 535–541.
[34] J. Ortega, M. Koppel, S. Argamon, Arbitrating among competing classifiers using learned referees, Knowl. Inf. Syst. 3 (2001) 470–490.
[35] H. Parvin, H. Alizadeh, M. Moshki, B. Minaei-Bidgoli, N. Mozayani, Divide & conquer classification and optimization by genetic algorithm, Proceedings of the Third International Conference on Convergence and Hybrid Information Technology (2008) 858–863 ICCIT'08. 2008.
[36] M.P. Perrone, Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure Optimization (1993).
[37] R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Mateo, CA, 1993.
[38] L. Rokach, Ensemble-based classifiers, Artif. Intell. Rev. 33 (2010) 1–39.
[39] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, R.C. Williamson, Estimating the support of a high-dimensional distribution, Neural Comput 13 (2001) 1443–1471.
[40] C. Seiffert, T.M. Khoshgoftaar, J. Van Hulse, A. Napolitano, RUSBoost: Improving classification performance when training data is skewed, in: Proceedings of the 19th International Conference on Pattern Recognition, ICPR 2008, 2008, pp. 1–4.
[41] P.K. Sharma, H. Haleem, T. Ahmad, Improving classification by outlier detection and removal, in: Proceedings of the49th Annual Convention of the Computer Society of India CSI: Emerging ICT for Bridging the Future, 2, 2015, pp. 621–628.
[42] J. Shawe-Taylor, S. Sun, A review of optimization methodologies in support vector machines, Neurocomputing. 74 (2011) 3609–3618.
[43] M.R. Smith, T. Martinez, Improving classification accuracy by identifying and removing instances that should be misclassified, in: International Joint Conference on Neural Networks (IJCNN), 2011, 2011, pp. 2690–2697.
[44] S.-H. Son, J.-Y. Kim, Data reduction for instance-based learning using entropy-based partitioning, Computer Science and its Applications, Springer, 2006, pp. 590–599.
[45] J. Song, H. Takakura, Y. Okabe, K. Nakao, Toward a more practical unsupervised anomaly detection system, Inf. Sci. 231 (2013) 4–14.
[46] S. Sun, Local within-class accuracies for weighting individual outputs in multiple classifier systems, Pattern Recognit. Lett. 31 (2010) 119–124.
[47] S. Sun, C. Zhang, Subspace ensembles for classification, Phys. Stat. Mech. Appl. 385 (2007) 199–207.
[48] G.I. Webb, Multiboosting: a technique for combining boosting and wagging, Mach. Learn. 40 (2000) 159–196.
[49] D.H. Wolpert, Stacked generalization, Neural Netw. 5 (1992) 241–259.
[50] M. Woźniak, M. Graña, E. Corchado, A survey of multiple classifier systems as hybrid systems, Inf. Fusion. 16 (2014) 3–17.
[51] Zhuowen Tu, Probabilistic boosting-tree: learning discriminative models for classification, recognition, and clustering, in: in: Proceedings of the Tenth IEEE International Conference on Computer Vision Vol. 1, IEEE, 2, 2005, pp. 1589–1596.