entropy

MDPI

Article

Decision Tree Integration Using Dynamic Regions of Competence

Jędrzej Biedrzycki and Robert Burduk *

Department of Systems and Computer Networks, Wroclaw University of Science and Technology, 50-370 Wroclaw, Poland; jedrzej.biedrzycki@pwr.edu.pl

* Correspondence: robert.burduk@pwr.edu.pl

Received: 10 September 2020; Accepted: 2 October 2020; Published: 5 October 2020

Abstract: A vital aspect of the Multiple Classifier Systems construction process is the base model integration. For example, the Random Forest approach used the majority voting rule to fuse the base classifiers obtained by bagging the training dataset. In this paper we propose the algorithm that uses partitioning the feature space whose split is determined by the decision rules of each decision tree node which is the base classification model. After dividing the feature space, the centroid of each new subspace is determined. This centroids are used in order to determine the weights needed in the integration phase based on the weighted majority voting rule. The proposal was compared with other Multiple Classifier Systems approaches. The experiments regarding multiple open-source benchmarking datasets demonstrate the effectiveness of our method. To discuss the results of our experiments, we use micro and macro-average classification performance measures.

Keywords: decision tree; random forest; majority voting; classifier ensemble; classifier integration

---

1. Introduction

Multiple Classifier Systems (MCS) are a popular approach to improve the possibilities of base classification models by building more stable and accurate classifiers [1]. MCS are one of the major development directions in machine learning [2,3]. MCS proved to have a significant impact on the system performance, therefore they are used in many practical aspects [4–7].

MCS are essentially composed of three stages: generation, selection and fussion or integration. The aim of the generation phase is to create basic classification models, which are assumed to be diverse. This goal is achieved, inter alia, by methods of dividing the feature space [8]. In the selection phase, one classifier (the classifier selection) or a certain subset of classifiers is selected (the ensemble pruning) learned at an earlier stage. The fusion or the integration process combines outputs of base classifiers to obtain an integrated model of classification, which is the final model of MCS. One of the commonly used methods to integrate base classifiers' outputs is the majority vote rule. In this method each base model has the same impact on the final decision of MCS. To improve the efficiency of MCS the weights are defined and used in the integration process. The use of weights allows to determine the influence of a particular base classifier on the final decision of MCS. The most commonly used approach to determining the weights uses probability error estimators or other factors [9–11]. A distance-weighted approach to calculating the weights is also often used in many problems, were the weights are determined [12–14]. In general, this approach is based on the query where the appropriate object is located. In this article, we use the feature subspace centroid in the definition of the distance-weighted approach.

There are, in general, two approaches to partition a dataset [15]. In horizontal partitioning the set of data instances is divided into a subset of datasets that are used to learn the base classifiers. Bagging bootstrap sampling to generate a training subset is one of the most used method in this type of datasets partitioning. In the vertical partitioning the feature set is divided into feature subsets that are used to learn the base classifiers. Based on vertical partitioning feature space the forest of decision trees was proposed in [16]. Contrary to the types of dataset partitioning mentioned above, the clustering and selection algorithm [17] is based on the clustering. After clustering, one classifier is selected for each feature subspace. In this algorithm the feature space partition is an independent process from the classifier selection process and precedes this selection. The non-sequential approach to clustering and selection algorithm was probesed in [18,19].

In this work, we propose a novel approach to determining the division feature space into the disjoint feature subspace. Contrary to the clustering and the selection method described above in our proposal the proces of partitioning the feature space follows base classifier learning. Additionally, the proposed approach does not use clustering to define a feature subspace. The partiotion of the feature space is defined by base classifier models, and exactly through their decision boundaries. According to our best knowledge the use of the decision boundary of base models for partitioning feature space is not represented in MCS. Finally, the centroids of proposed feature subspace are used in the weighted majority voting rule to define the final MCS decision.

Given the above, the main objectives of this work can be summarized as follows:

- A proposal of a new partitioning of the feature space whose split is determined by the decision bonduaries of each decision tree node which is a base classification model.
- The proposal of a new weighted majority voting rule algorithm dedicated to the fusion of decision tree models.
- An experimental setup to compare the proposed method with other MCS approaches using different performance measures.

The outline of the paper is as follows: In Section 2 related works are presented. Section 3 presents the proposed approach to MCS fusion process. In Section 4 the experiments that were carried out and the discusion of the obtained results are presented. Finally, we conclude the paper in Section 6.

2. Related works

Classifier integration using the geometrical representation has already been mentioned in [20]. Based on transformations in the geometrical space spread on real-valued, non-categorical features this procedure has proven itself to be more effective in comparison to others, commonly used integration techniques such as majority voting [21]. The authors have studied and proved the effectiveness of an integration algorithm based on averaging and taking median of values of the decision boundary in the SVM classifiers [22]. Next, two algorithms for decision trees were proposed and evaluated [23,24]. They have proven themselves to provide better classification quality and ease of use than referential methods.

Polianskii and Pokorny have examined a geometric approach to the classification using Voronoi cells [25]. Voronoi cells fulfill the role of the atomic elements being classified. Labels of the nearest training objects are assigned to the boundaries. The algorithms walks along the boundaries and integrates them with respect to the associated class. SVM, NN and random forest classifiers were used in evaluation.

The nearest neighbor algorithm can be used to test which Voronoi cell an object belongs to [26]. By avoiding the calculation of the Voronoi cells geometry, the test appears to be very efficient. On that basis a search lookup was described by Kushilevitz et al. [27]. A space-efficient data structure is utilized to find an approximately nearest neighbor in nearly-quadratic time with respect to the dimensionality.

However, the nearest neighbor algorithms are difficult. The number of prototypes needs to be specified beforehand. Using too many causes high computational complexity. Too few, on the other hand, can result in an oversimplified classification model. This matters especially when datasets are not linearly separable, have island–shaped decision space, etc. There are several ways to solve this problem that can be found in the literature. Applying Generalized Condensed Nearest Neighbor rule to obtain a set of prototypes is one of the possible solutions [28]. In this method a constraint is added, that each of the prototypes has to come from the training dataset. A different approach was proposed by Gou et al. [29]. Firstly, kNN algorithm is used to obtain a certain number of prototypes for each class. Afterwards the prototypes obtained in the first step are transformed by the local mean vectors. This results in a better representation of the distribution of the decision space.

Decision trees are broadly used due to their simplicity, intuitive approach and at the same time good efficiency. The way they classify the objects is by recursive partitioning of the classification space [30]. Although they have first appeared more than three decades ago [31], the decision tree algorithm and its derivatives are in use in a range of industry branches [32].

At some point it has been noticed that the local quality of each of the base classifiers is different. The classifier selection process was introduced in order to choose a subset of base classifiers that have the best classification quality over the region. The selection is called static, when the division can be determined prior to the classification. In the opposite scenario the new pattern is used to test the models' quality [33]. Kim and Ko [34] have shown a greater improvement in the classification when using local confidence over averaging over the entire decision space.

Another approach to the classifier integration is by using a combination of weighting and local confidence estimation [35]. The authors noticed, that using only a subset of points limited to a certain area in the training process results in a better classification performance.

An article [36] discusses a variation of the majority voting technique. A probability estimate is computed as the ratio of properly classified validation objects over certain geometric constraints known a priori. Regions that are functionally independent from each other are treated separately. The proposed approach provides a significant improvement in the classification quality. The downside of this method is that the knowledge of the domain is necessary to provide a proper division. Additionally, the split of the classification space has to be done manually. The performance of the algorithm was evaluated using a retinal image and classification in its anatomic regions.

An improvement in the weighted majority voting classification can be observed for class–wise approach covered in [37]. For each label weights are determined separately for the objects in the validation dataset.

Random forest, introduced by Breiman in 2001 [38] is one of the most popular ensemble methods. It has proven itself to be very effective and many related algorithms were developed since then. Fernandez et al. studied 179 different classifications algorithms using 121 datasets [39]. The random forest outperforms most of the examined classifiers. It uses decision trees trained on distinct subsets of the training dataset. A majority voting over classifications of every model for an object under test is calculated as the final result.

Numerous algorithms involving gradient boosting and decision trees have emerged. Extreme Gradient Boosting (XGB) is an implementation of one of the most widely spread stacking techniques. It is used especially in machine learning competitions [40–42]. In theory subsequent decision trees are trained. Consecutive models minimize the value of loss function left by their predecessors [43]. Another implementation of Gradient Boosting Decision Tree designed with performance in mind, especially when working with datasets with many dimensions, is LightGBM [44]. Compared to the previous library, statistically no loss of performance in the classification is observed, but the process of training can be up to 20 times faster.

Vertical or horizontal partitioning can be used to force the diversity between base classifiers [30]. The datasets of extreme sizes are classified better using horizontal partitioning compared to bagging, boosting or other ensemble techniques [45].

3. Proposed Method

The proposed method is based on previous works of authors, but suggests a slightly different approach [23,24]. While the cited articles used static division into regions of competence, this paper presents an algorithm with a dynamic approach. The main goal of introducing the dynamically generated Voronoi cells is to achieve better performance than with other referential methods of the decision tree commitee ensembling: majority voting and random forest.

Before proceeding with the algorithm, datasets are normalized to the unit cube (every feature takes values in range of $[0, 1]$) and two most informative features are extracted. Feature extraction is conducted using ANOVA method.

The first step of the presented algorithm is training a pool of base decision trees. To make sure the classifiers are different from one another, they are trained on the random subsets of the dataset. Having a commitee of decision trees trained, we are extracting rectangular regions that fulfill the following properties:

- Their area is maximal.
- Every point they span is labeled with the same label by every single classifier (labels can differ across different classifiers). In other words regions span over the area of objects equally labelled by the classifier points.

In practice this means, that the entire space is divided along every dimension at all the split points of every decision tree. This way the regions are of the same class as indicated by every model.

Having the space divided into subspaces, midpoints are calculated. Let us denote by S the set of obtained subspaces and by $(x_{s,1,min}; x_{s,1,max})$ and $(x_{s,2,min}, x_{s,2,max})$ the range of subspace s along axis x_1 and x_2 respectively. The midpoint of subspace s will be denoted as $x_{s,mid}$. For every subspace and every label the weight is calculated using the following formula:

$$f(\Psi_i, s_0) = \frac{1}{\sigma} \sum_{s \in S} c_{s,\Psi_i} (1 - d(x_{s_0,mid}, x_{s,mid}))\delta(s_0, s) + \frac{c_{s_0,\Psi_i}}{2n} \tag{1}$$

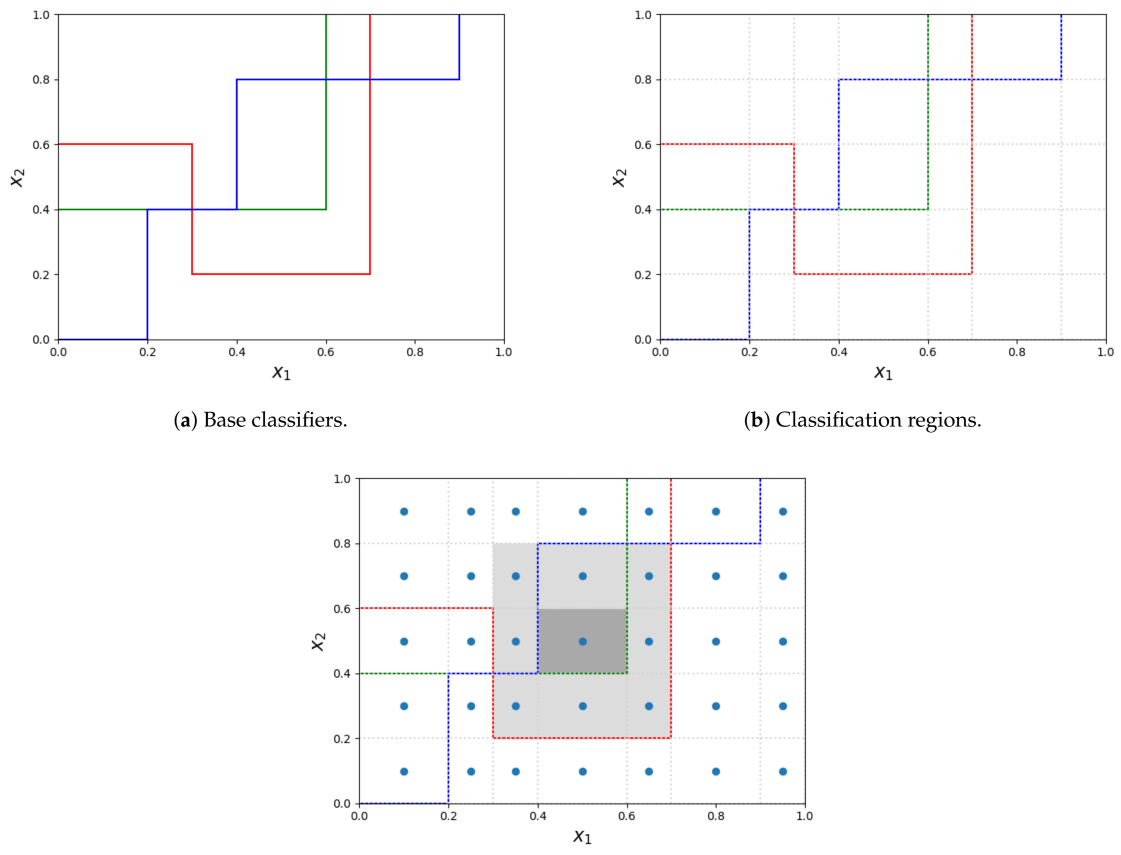
where $d(p_1, p_2)$ is the euclidean distance between the points p_1 and p_2, c_{s,Ψ_i} is the number of classifiers that classify the subspace s with the label Ψ_i, σ is the correction which purpose is to make the sum of weights equal 1 and $\delta(s_0, s)$ is a function that returns 1 if s_0 and s are neighbors and 0 otherwise, i.e.,

$$\delta(s_0, s) = \begin{cases} 1 & \text{if } x_{s_0,1,min} = x_{s,2,max} \text{ or } x_{s_0,1,max} = x_{s,1,min} \\ & \text{or } x_{s_0,2,min} = x_{s,2,max} \text{ or } x_{s_0,2,max} = x_{s,2,min} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

It's important to notice, that according to the Formula (2), $\delta(s, s) = 0$ for every subspace s. This is because the contribution of the subspace itself is reflected by the second summand of the Equation (1). The term $2n$ was chosen in the denominator, because then the subspace s_0 makes up half of the weight's value, i.e.,

$$\sum_{i=1}^{n} \frac{c_{s_0,\Psi_i}}{2n} = \frac{1}{2}$$

The process of obtaining subspaces is depicted in the Figure 1. Let us suppose, that all the base decision trees (colorful lines on subfigure a) are oriented in the same way - all the points below the decision boundary are classified by the given decision tree with a single label, different from all the objects above the line. As it was stated before, the competence regions are obtained by splitting the entire space at the splitpoints of all the decision trees (subfigure b). When calculating the weight of the label for each region, the region itself (filled with dark grey in subfigure c) together with its neighbors (lightgrey in subfigure c) are considered. Whereas the region itself contributes to half of its weight, contributions from every neighbor depend on the distance between its midpoint and the midpoint of the considered region. The entire procedure is presented in Algorithm 1.



(**a**) Base classifiers.

(**b**) Classification regions.

(**c**) Specific subspace (darkgrey) with its neighbors (lightgrey). Midpoints are designated with blue dots.

Figure 1. The process of extracting subspaces from base classifiers and determining neighbors for a subspace.

Algorithm 1: Classification algorithm using dynamic regions of competence obtained from decision trees.

```
    Input   : $K$ – number of base classifiers ($\Psi_1, \Psi_2, \ldots, \Psi_K$)
    Output: Integrated decision tree $\Psi_i$
1  Normalize the dataset and select two most informative features.
2  Split dataset into $K + 1$ subsets ($K$ for training every base decision tree and 1 for testing).
3  Train base classifiers $\Psi_1, \Psi_2, \ldots, \Psi_K$ and obtain their geometrical representation (splits
     and labels).
4  Divide the feature space using splits of all the decision trees.
5  For every region and every label calculate the weight using formula (1).
6  Classify every region by picking the label with the highest weight value.
```

4. Experimental Setup

The algorithm was implemented in Scala. Decision tree and random forest implementation from Spark MlLib were used [46]. The statistical analysis was performed with Python and libraries Numpy, Scipy and Pandas [47–49]. Matplotlib was used for plotting [50]. In Spark's implementation the bottommost elements (leaves) are classified with a single label. The algorithm performs a greedy, recursive partitioning in order to maximize the information gain in every tree node. Gini impurity is used as the homogeneity measure. Continuous feature discretization is conducted using 32 bins.

The source code used to conduct experiments is available online (https://github.com/TAndronicus/dynamic-dtree).

The experiments were conducted using open-source benchmarking datasets from repositories UCI and KEEL [51,52]. Table 1 describes the datasets used with the number of features, instances and imbalance ratio.

Table 1. Descriptions of datasets used in experiments (name with abbreviation, number of instances, number of features, imbalance ratio).

Dataset	#inst	#f	Imb
Indoor Channel Measurements (aa)	7840	5	208.0
Appendicitis (ap)	106	7	4.0
Banana (ba)	5300	2	5.9
QSAR biodegradation (bi)	1055	41	2.0
Liver Disorders (BUPA) (bu)	345	6	1.4
Cryotherapy (c)	90	7	1.1
Banknote authentication (d)	1372	5	1.2
Ecoli (e)	336	7	71.5
Haberman's Survival (h)	306	3	2.8
Ionosphere (io)	351	34	1.8
Iris plants (ir)	150	4	1.0
Magic (ma)	19,020	10	1.0
Ultrasonic flowmeter diagnostics (me)	540	173	1.4
Phoneme (ph)	5404	5	2.4
Pima (pi)	768	8	1.9
Climate model simulation crashes (po)	540	18	10.7
Ring (r)	7400	20	1.0
Spambase (sb)	4597	57	1.5
Seismic-bumps (se)	2584	19	14.2
Texture (te)	5500	40	1.0
Thyroid (th)	7200	21	1.0
Titanic (ti)	2201	3	2.1
Twonorm (tw)	7400	20	1.0
Breast Cancer (Diagnostic) (wd)	569	30	1.7
Breast Cancer (Original) (wi)	699	9	1.9
Wine quality – red (wr)	1599	11	68.1
Wine quality – white (ww)	4898	11	439.6
Yeast (y)	1484	8	92.6

The imbalance ratio was given to stress the fact that accuracy is not a reliable metric when comparing the performance of the presented algorithm and the reference. It is calculated as the quotient of the count of objects with the major label (most frequent) and the objects with minor label (least common): $\text{Imb} = \frac{\#\text{major class objects}}{\#\text{minor class objects}}$ [53]. If the value of Imb equals 1, then the dataset is balanced—all classes have the same amount of instances. The larger the value, the more imbalanced the dataset is. Some of the datasets are highly imbalanced, because of the low imbalance ratio, so other metrics other than average accuracy should be considered when comparing the performance of classifiers. The reason is explained in the following example. Suppose $\text{Imb} = 9$ for a binary classification problem. When a classifier labels all the test objects with the label of the major class, its accuracy is $\text{ACC} = \frac{9}{9+1} = 90\%$. In the parentheses, together with the names, abbreviations of the datasets names were placed by which they will be further referenced for brevity.

The experiments were conducted according to the procedure described in Section 3 and repeated 10 times for each hyperparameter set. Together with integrated classifiers, referential methods were evaluated: majority voting of the base classifiers and random forest. The results were averaged. $K = 3$ was taken as the number of base classifiers.

5. Results

The purpose of the experiments was to compare the classification performance measures obtained by the proposed algorithm (with the subscript i) with the known methods as references: majority voting (subscript mv) and random forest (subscript rf). The experiments were conducted 10 times for each setup and the results were averaged. Because we conducted experiments on the multiclass datasets, as the classification evaluation metrics we use micro- and macro-average precision, recall and F-score which is the harmonic mean of precision and recall. For this reason F-score takes both false positives and false negatives into account. Additionally, we present the results for overall accuracy. The F-score was computed alongside the accuracy because of the high imbalance of multiple datasets used, as it was indicated in Section4. The F-score describes the quality of a classifier much better than the overall accuracy for the datasets with a high imbalance ratio and gives a better performance measure of the incorrectly classified cases than the overall accuracy. Accuracy can be in this case artificially high. The metrics are calculated as defined in [54]. In the Table 2 the results are gathered: average accuracy, micro- and macro-average F-score, while the Tables 3 and 4 show results for micro- and macro-average respectively. Together with the mentioned metrics, Friedman ranks are presented in the last row – the smaller the rank, the better the classifier performs. However, it should be noted that for micro-average performance measures the result obtained for precision and recall are the same. This result is justified by the micro-averaging disadvantage, because for the frequent single-label per instance problems Precision$_\mu$ = Recall$_\mu$ [55].

Table 2. Average accuracy and f-scores for the random forest, the majority voting and the proposed algorithm together with Friedman ranks.

	Average Accuracy			**F-Score$_\mu$**			**F-Score$_M$**		
Dataset	Ψ_{mv}	Ψ_{rf}	Ψ_i	Ψ_{mv}	Ψ_{rf}	Ψ_i	Ψ_{mv}	Ψ_{rf}	Ψ_i
aa	0.917	0.918	0.919	0.469	0.474	0.477	0.196	0.192	0.176
ap	0.853	0.812	0.863	0.853	0.812	0.863	0.676	0.559	0.692
ba	0.789	0.808	0.815	0.683	0.712	0.722	0.483	0.493	0.502
bi	0.736	0.736	0.702	0.736	0.736	0.702	0.717	0.717	0.561
bu	0.579	0.527	0.536	0.579	0.527	0.536	0.563	0.520	0.512
c	0.762	0.867	0.684	0.762	0.867	0.684	0.773	0.870	0.698
d	0.935	0.935	0.938	0.935	0.935	0.938	0.934	0.934	0.936
e	0.825	0.827	0.825	0.414	0.423	0.414	0.110	0.167	0.106
h	0.637	0.691	0.657	0.637	0.691	0.657	0.480	0.581	0.491
io	0.862	0.868	0.458	0.862	0.868	0.458	0.845	0.853	0.578
ir	0.965	0.961	0.978	0.947	0.942	0.968	0.945	0.943	0.968
ma	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
me	0.582	0.681	0.615	0.582	0.681	0.615	0.583	0.688	0.607
ph	0.771	0.767	0.774	0.771	0.767	0.774	0.720	0.718	0.724
pi	0.699	0.685	0.704	0.699	0.685	0.704	0.661	0.653	0.670
po	0.879	0.872	0.897	0.879	0.872	0.897	0.468	0.466	0.473
r	0.726	0.723	0.728	0.726	0.723	0.728	0.733	0.730	0.735
sb	0.711	0.718	0.712	0.711	0.718	0.712	0.686	0.694	0.686
se	0.924	0.921	0.926	0.924	0.921	0.926	0.520	0.516	0.497
te	0.889	0.890	0.892	0.389	0.392	0.408	0.393	0.387	0.404
th	0.983	0.981	0.982	0.974	0.972	0.973	0.849	0.825	0.851
ti	0.788	0.778	0.681	0.788	0.778	0.681	0.752	0.732	0.405
tw	0.717	0.714	0.724	0.717	0.714	0.724	0.717	0.714	0.724
wd	0.902	0.893	0.918	0.902	0.893	0.918	0.893	0.884	0.911
wi	0.936	0.955	0.944	0.936	0.955	0.944	0.931	0.951	0.941
wr	0.831	0.827	0.823	0.493	0.481	0.468	0.241	0.227	0.225
ww	0.839	0.838	0.840	0.459	0.457	0.464	0.205	0.224	0.208
y	0.866	0.861	0.865	0.349	0.325	0.344	0.223	0.214	0.234
rank	2.00	2.14	1.61	2.00	2.14	1.61	1.93	2.04	1.79

Table 3. Micro-average precision and recall for the random forest, the majority voting and the proposed algorithm together with Friedman ranks.

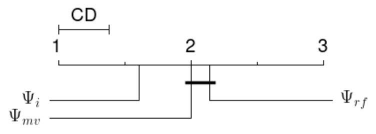
	Precision$_\mu$			**Recall$_\mu$**		
Dataset	Ψ_{mv}	Ψ_{rf}	Ψ_i	Ψ_{mv}	Ψ_{rf}	Ψ_i
aa	0.469	0.475	0.477	0.469	0.473	0.477
ap	0.853	0.812	0.863	0.853	0.812	0.863
ba	0.683	0.712	0.722	0.683	0.712	0.722
bi	0.736	0.736	0.702	0.736	0.736	0.702
bu	0.579	0.527	0.536	0.579	0.527	0.536
c	0.762	0.867	0.684	0.762	0.867	0.684
d	0.935	0.935	0.938	0.935	0.935	0.938
e	0.418	0.424	0.417	0.411	0.423	0.411
h	0.637	0.691	0.657	0.637	0.691	0.657
io	0.862	0.868	0.458	0.862	0.868	0.458
ir	0.947	0.942	0.968	0.947	0.942	0.968
ma	1.000	1.000	1.000	1.000	1.000	1.000
me	0.582	0.681	0.615	0.582	0.681	0.615
ph	0.771	0.767	0.774	0.771	0.767	0.774
pi	0.699	0.685	0.704	0.699	0.685	0.704
po	0.879	0.872	0.897	0.879	0.872	0.897
r	0.726	0.723	0.728	0.726	0.723	0.728
sb	0.711	0.718	0.712	0.711	0.718	0.712
se	0.924	0.921	0.926	0.924	0.921	0.926
te	0.389	0.392	0.408	0.389	0.392	0.408
th	0.974	0.972	0.973	0.974	0.972	0.973
ti	0.788	0.778	0.681	0.788	0.778	0.681
tw	0.717	0.714	0.724	0.717	0.714	0.724
wd	0.902	0.893	0.918	0.902	0.893	0.918
wi	0.936	0.955	0.944	0.936	0.955	0.944
wr	0.493	0.481	0.468	0.493	0.481	0.468
ww	0.459	0.457	0.464	0.459	0.457	0.464
y	0.350	0.325	0.344	0.349	0.325	0.344
rank	2.00	2.14	1.64	2.00	2.14	1.61

Table 4. Macro-average precision and recall for the random forest, the majority voting and the proposed algorithm together with Friedman ranks.

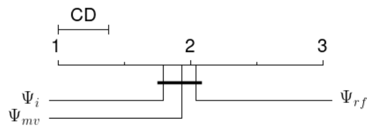
	Precision$_M$			**Recall$_M$**		
Dataset	Ψ_{mv}	Ψ_{rf}	Ψ_i	Ψ_{mv}	Ψ_{rf}	Ψ_i
aa	0.179	0.171	0.152	0.217	0.218	0.209
ap	0.705	0.557	0.710	0.663	0.563	0.684
ba	0.475	0.475	0.486	0.491	0.512	0.519
bi	0.712	0.712	0.526	0.721	0.721	0.614
bu	0.564	0.521	0.513	0.561	0.520	0.511
c	0.779	0.872	0.702	0.767	0.868	0.693
d	0.936	0.935	0.938	0.932	0.933	0.935
e	0.079	0.123	0.075	0.182	0.259	0.186
h	0.474	0.594	0.485	0.486	0.568	0.500
io	0.860	0.881	0.596	0.831	0.828	0.562
ir	0.944	0.942	0.968	0.945	0.944	0.968
ma	1.000	1.000	1.000	1.000	1.000	1.000
me	0.582	0.683	0.614	0.585	0.692	0.600
ph	0.726	0.721	0.730	0.715	0.716	0.718
pi	0.664	0.652	0.670	0.659	0.655	0.669
po	0.451	0.450	0.451	0.487	0.483	0.496
r	0.742	0.738	0.743	0.724	0.721	0.726
sb	0.723	0.728	0.721	0.653	0.663	0.655
se	0.542	0.527	0.495	0.504	0.507	0.501
te	0.397	0.380	0.400	0.390	0.393	0.409
th	0.816	0.803	0.826	0.886	0.848	0.879
ti	0.853	0.780	0.341	0.673	0.692	0.500
tw	0.718	0.714	0.724	0.717	0.714	0.724
wd	0.894	0.883	0.911	0.892	0.886	0.911
wi	0.926	0.948	0.931	0.935	0.954	0.951
wr	0.246	0.228	0.234	0.236	0.226	0.216
ww	0.226	0.247	0.230	0.189	0.206	0.191
y	0.232	0.200	0.244	0.216	0.230	0.226
rank	1.86	2.07	1.79	2.18	1.82	1.82

6. Discussion

For the proposed weighting method of the decision tree integration all the calculated classification performance measures are better than of the referrential algorithms as indicated by the Friedman ranks. This statement holds true for all the classification performance measures that have been used. Post-hoc Nemenyi test after Friedman ranking requires the difference in ranks of 0.38 to define a significant statistical difference between the algorithms. For F-score$_\mu$ performance measure this condition is met, which means that the proposed method Ψ_i achieves statistically better results than the reference methods Ψ_{rf} and Ψ_{mv}. Whereas for F-score$_M$ performacne measure there is no such property as shown in the Figure 2. The micro-average measure counts the fraction of instances predicted correctly across all classes. For this reason the micro-average can be a more useful metric than macro-average in the class imbalance dataset. Thus, the results show that the proposed method improves the classification results, in particular of imbalanced datasets. This conclusion is confirmed by the values obtained for other performance measures. And so for micro-avarage precision and recall the difference in ranks betwenn Ψ_i and Ψ_{rf} indicated the statistical differences in the results. The corresponding difference for Ψ_i and Ψ_{rf} algorithms is very close to being able to state the statistical differences in the results because it is equal 0.36 (see Table 3). In case of macro-avarage precision the obtained results do not indicate significant difference in this performance measure. Whereas for macro-avarage recall (see Table 4) the obtained avarage Friedman ranks are equal for Ψ_i and Ψ_{rf} algorithms.



(**a**) Friedman ranks with the critical Nemenyi test value for average accuracy and F-score$_\mu$.



(**b**) Friedman ranks with the critical Nemenyi test value for F-score$_M$.

Figure 2. Friedman ranks for calculated metrics together with Nemenyi critical values.

7. Conclusions

This paper presents a new approach for determining MCS. Contrary to the clustering and selection method we propose that the feature space partition is based on decision bonduaries defined by base classifier models. It means that we propose to use learned base classification models instead of clustering to determine the feature subspace. The centroids of the proposed feature subspace are used in the weighted majority voting rule to define the final MCS decision. In particular, a class label prediction for each feature subspace is based on adjacent feature subspaces.

The experimental results show that the proposed method may create an ensemble classifier that outperforms the commonly used methods of combining decision tree models—the majority voting and RF. Especially, the results show that the proposed method statistically improves the classification results measured by the mico-avarage F_1 classification performance measure.

According to our best knowledge the use of the decision boundary of base models to partition feature space is not represented in MCS. On the other hand, the proposed approach has also a drawback, because it uses geometrical centroids of defined feature subspaces. Consequently, our future research needs to be aimed at finding centroids of objects belonging to particular feature subspaces. Additionally, we can consider another neighborhood of a given feature subspace necessary to determine the decision rule. This neighborhood may, for example, depend on the number of objects in particular the feature subspace.

Author Contributions: Conceptualization, R.B. and J.B.; methodology, R.B.; software, J.B.; validation, R.B. and J.B.; formal analysis, J.B.; investigation, J.B.; resources, R.B.; data curation, J.B.; writing—original draft preparation, J.B.; writing—review and editing, R.B. and J.B.; visualization, R.B. and J.B.; supervision, R.B.; project administration, R.B.; funding acquisition, R.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Science Centre, Poland under the grant no. 2017/25/B/ST6/01750.

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

Abbreviations

The following abbreviations are used in this manuscript:

MV	Majority Voting
RF	Random Forest
SVM	Support Vector Machine

References

1. Sagi, O.; Rokach, L. Ensemble learning: A survey. *Wiley Interdiscip. Rev.-Data Mining Knowl. Discov.* **2018**, *8*, e1249. [CrossRef]
2. Alpaydin, E. *Introduction to Machine Learning*; MIT Press: Cambridge, MA, USA, 2020.
3. Andrysiak, T. Machine learning techniques applied to data analysis and anomaly detection in ECG signals. *Appl. Artif. Intell.* **2016**, *30*, 610–634. [CrossRef]
4. Burduk, A.; Grzybowska, K.; Safonyk, A. The Use of a Hybrid Model of the Expert System for Assessing the Potentiality Manufacturing the Assumed Quantity of Wire Harnesses. *LogForum* **2019**, *15*, 459–473. [CrossRef]
5. Dutta, V.; Choraś, M.; Pawlicki, M.; Kozik, R. A Deep Learning Ensemble for Network Anomaly and Cyber-Attack Detection. *Sensors* **2020**, *20*, 4583. [CrossRef]
6. Heda, P.; Rojek, I.; Burduk, R. Dynamic Ensemble Selection–Application to Classification of Cutting Tools. In *International Conference on Computer Information Systems and Industrial Management;* Springer: Berlin/Heidelberg, Germany, 2020; pp. 345–354.
7. Xiao, J. SVM and KNN ensemble learning for traffic incident detection. *Physica A* **2019**, *517*, 29–35. [CrossRef]
8. Rokach, L. Decomposition methodology for classification tasks: A meta decomposer framework. *Pattern Anal. Appl.* **2006**, *9*, 257–271. [CrossRef]
9. Burduk, R. Classifier fusion with interval-valued weights. *Pattern Recognit. Lett.* **2013**, *34*, 1623–1629. [CrossRef]
10. Mao, S.; Jiao, L.; Xiong, L.; Gou, S.; Chen, B.; Yeung, S.K. Weighted classifier ensemble based on quadratic form. *Pattern Recognit.* **2015**, *48*, 1688–1706. [CrossRef]
11. Woźniak, M.; Graña, M.; Corchado, E. A survey of multiple classifier systems as hybrid systems. *Inf. Fusion* **2014**, *16*, 3–17. [CrossRef]
12. Aguilera, J.; González, L.C.; Montes-y Gómez, M.; Rosso, P. A new weighted k-nearest neighbor algorithm based on newton's gravitational force. In *Iberoamerican Congress on Pattern Recognition;* Springer: Berlin/Heidelberg, Germany, 2018; pp. 305–313.
13. Ksieniewicz, P.; Burduk, R. Clustering and Weighted Scoring in Geometric Space Support Vector Machine Ensemble for Highly Imbalanced Data Classification. In *International Conference on Computational Science*; Springer: Berlin/Heidelberg, Germany, 2020, pp. 128–140.
14. Geler, Z.; Kurbalija, V.; Ivanović, M.; Radovanović, M. Weighted kNN and constrained elastic distances for time-series classification. *Expert Syst. Appl.* **2020**, 113829.
15. Guggari, S.; Kadappa, V.; Umadevi, V. Non-sequential partitioning approaches to decision tree classifier. *Future Computing Inform. J.* **2018**, *3*, 275–285. [CrossRef]
16. Ho, T.K. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.* **1998**, *20*, 832–844.
17. Kuncheva, L.I. Clustering-and-selection model for classifier combination. In *KES'2000. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies. Proceedings (Cat. No. 00TH8516)*; IEEE: Piscataway, NJ, USA, 2000; Volume 1, pp. 185–188.
18. Jackowski, K.; Wozniak, M. Algorithm of designing compound recognition system on the basis of combining classifiers with simultaneous splitting feature space into competence areas. *Pattern Anal. Appl.* **2009**, *12*, 415. [CrossRef]

19. Lopez-Garcia, P.; Masegosa, A.D.; Osaba, E.; Onieva, E.; Perallos, A. Ensemble classification for imbalanced data based on feature space partitioning and hybrid metaheuristics. *Appl. Intell.* **2019**, *49*, 2807–2822. [CrossRef]
20. Pujol, O.; Masip, D. Geometry-Based Ensembles: Toward a Structural Characterization of the Classification Boundary. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 1140–1146. [CrossRef]
21. Burduk, R. Integration Base Classifiers in Geometry Space by Harmonic Mean. In *Artificial Intelligence and Soft Computing*; Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M., Eds.; Springer: Berlin/Heidelberg, Germany, 2018; pp. 585–592.
22. Burduk, R.; Biedrzycki, J. Integration and Selection of Linear SVM Classifiers in Geometric Space. *J. Univers. Comput. Sci.* **2019**, *25*, 718–730.
23. Biedrzycki, J.; Burduk, R. Integration of decision trees using distance to centroid and to decision boundary. *J. Univers. Comput. Sci.* **2020**, *26*, 720–733.
24. Biedrzycki, J.; Burduk, R. Weighted scoring in geometric space for decision tree ensemble. *IEEE Access* **2020**, *8*, 82100 – 82107. [CrossRef]
25. Polianskii, V.; Pokorny, F.T. Voronoi Boundary Classification: A High-Dimensional Geometric Approach via Weighted M onte C arlo Integration. In *International Conference on Machine Learning*; Omnipress: Madison, WI, USA, 2019; pp. 5162–5170.
26. Biau, G.; Devroye, L. *Lectures on the Nearest Neighbor Method*; Springer: Berlin/Heidelberg, Germany, 2015.
27. Kushilevitz, E.; Ostrovsky, R.; Rabani, Y. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.* **2000**, *30*, 457–474. [CrossRef]
28. Kheradpisheh, S.R.; Behjati-Ardakani, F.; Ebrahimpour, R. Combining classifiers using nearest decision prototypes. *Appl. Soft. Comput.* **2013**, *13*, 4570–4578. [CrossRef]
29. Gou, J.; Zhan, Y.; Rao, Y.; Shen, X.; Wang, X.; He, W. Improved pseudo nearest neighbor classification. *Knowl.-Based Syst.* **2014**, *70*, 361–375. [CrossRef]
30. Rokach, L. Decision forest: Twenty years of research. *Inf. Fusion* **2016**, *27*, 111–125. [CrossRef]
31. Quinlan, J.R. Induction of Decision Trees. *Mach. Learn.* **1986**, *1*, 81–106. [CrossRef]
32. Tan, P.N.; Steinbach, M.M.; Kumar, V. *Introduction to Data Mining*; Addison-Wesley: Boston, MA, USA, 2005.
33. Ponti, M.P. Combining Classifiers: From the Creation of Ensembles to the Decision Fusion. In Proceedings of the 2011 24th SIBGRAPI Conference on Graphics, Patterns, and Images Tutorials, Alagoas, Brazil, 28–30 August 2011; pp. 1–10.
34. Kim, E.; Ko, J. Dynamic Classifier Integration Method. In *Multiple Classifier Systems*; Oza, N.C., Polikar, R., Kittler, J., Roli, F., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 97–107.
35. Sultan Zia, M.; Hussain, M.; Arfan Jaffar, M. A novel spontaneous facial expression recognition using dynamically weighted majority voting based ensemble classifier. *Multimed. Tools Appl.* **2018**, *77*, 25537–25567. [CrossRef]
36. Hajdu, A.; Hajdu, L.; Jonas, A.; Kovacs, L.; Toman, H. Generalizing the majority voting scheme to spatially constrained voting. *IEEE Trans. Image Process.* **2013**, *22*, 4182–4194. [CrossRef] [PubMed]
37. Rahman, A.F.R.; Alam, H.; Fairhurst, M.C. Multiple Classifier Combination for Character Recognition: Revisiting the Majority Voting System and Its Variations. In *Document Analysis Systems V*; Lopresti, D., Hu, J., Kashi, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2002; pp. 167–178.
38. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
39. Fernández-Delgado, M.; Cernadas, E.; Barro, S.; Amorim, D. Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.* **2014**, *15*, 3133–3181.
40. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*; Association for Computing Machinery: New York, NY, USA, 2016; pp. 785–794.
41. Taieb, S.B.; Hyndman, R.J. A gradient boosting approach to the Kaggle load forecasting competition. *Int. J. Forecast.* **2014**, *30*, 382–394. [CrossRef]
42. Sheridan, R.P.; Wang, W.M.; Liaw, A.; Ma, J.; Gifford, E.M. Extreme Gradient Boosting as a Method for Quantitative Structure-Activity Relationships. *J. Chem Inf. Model.* **2016**, *56*, 2353–2360. [CrossRef]
43. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]

44. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.Y. LightGBM: A highly efficient gradient boosting decision tree. In *NIPS'17 Proceedings of the 31st International Conference on Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 3149–3157.
45. Chawla, N.V.; Hall, L.O.; Bowyer, K.W.; Kegelmeyer, W.P. Learning Ensembles from Bites: A Scalable and Accurate Approach. *J. Mach. Learn. Res.* **2004**, *5*, 421–451.
46. Meng, X.; Bradley, J.; Yavuz, B.; Sparks, E.; Venkataraman, S.; Liu, D.; Freeman, J.; Tsai, D.; Amde, M.; Owen, S.; et al. MLlib: Machine Learning in Apache Spark. *J. Mach. Learn. Res.* **2015**, *17*, 1235–1241.
47. Oliphant, T. *NumPy: A guide to NumPy*; Trelgol Publishing: Spanish Fork, UT, USA, 2006.
48. Jones, E.; Oliphant, T.; Peterson, P. SciPy: Open Source Scientific Tools for Python. Available online: https://www.mendeley.com/catalogue/cc1d80ce-06d6-3fc5-a6cf-323eaa234d84/ (accessed on 20 September 2020).
49. McKinney, W. Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference, Austin, TX, USA, 28 June–3 July 2020; van der Walt, S., Millman, J., Eds.; pp. 56–61. [CrossRef]
50. Hunter, J.D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* **2007**, *9*, 90–95. [CrossRef]
51. Dua, D.; Graff, C. UCI Machine Learning Repository. 2017. Available online: https://ergodicity.net/2013/07/ (accessed on 20 September 2020).
52. Alcalá-Fdez, J.; Fernández, A.; Luengo, J.; Derrac, J.; García, S.; Sánchez, L.; Herrera, F. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *J. Mult.-Valued Log. Soft Comput.* **2011**, *17*.
53. Ortigosa-Hernández, J.; Inza, I.; Lozano, J.A. Measuring the class-imbalance extent of multi-class problems. *Pattern Recognit. Lett.* **2017**, *98*, 32–38. [CrossRef]
54. Sokolova, M.; Lapalme, G. A systematic analysis of performance measures for classification tasks. *Inform. Process Manag.* **2009**, *45*, 427–437. [CrossRef]
55. Van Asch, V. *Macro- and Micro-Averaged Evaluation Measures*; Basic Draft; CLiPS: Antwerpen, Belgium, 2013; Volume 49.