WROCŁAW UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF ELECTRONICS

PhD Thesis

Integration of decision trees in geometric space

AUTHOR:
Jędrzej Biedrzycki

SUPERVISOR:
Ph.D., D.Sc. Robert Burduk

WROCŁAW, 2021

I would like to dedicate this work to my wonderful girlfriend, Asia, for her unmeasurable patience and understanding. Without her strong support, I would not be able to fully pursue my goals.

I am very grateful to my parents for their invaluable contribution to my education throughout the years.

Contents

Index of figures

Index of tables

Abbreviations

ACC – **Acc**uracy – ratio of properly classified test objects

ANN – **A**rtificial **n**eural **n**etwork – artificial intelligence model composed of layers of artificial neurons

ANOVA – **An**alysis **o**f **va**riance – form of statistical hypothesis testing

API – **A**pplication **p**rogramming **i**nterface – interface defining interactions with the software

AUC – **a**rea **u**nder **c**urve – definite integral of the curve, in this case the curve is ROC – receiver operating characteristic

CAD – **C**omputer **a**ided **d**iagnosis – intelligent system assisting doctors in the interpretation of medical data

EoC – **E**nsemble **o**f **c**lassifiers – integrated classifier, a classification model built on top of base classifiers

kNN – **k**-**n**earest **n**eighbors – classification algorithm assigning a label based on the plurality voting rule of the k closest training objects

MCS – **M**ulti**c**lassifier **s**ystem – integrated classifier, see EoC

MV – **M**ajority **v**oting – classification rule used in integrated classifiers, assigning a label to the object under test as the predictions of the majority of base models

NN – **N**earest **n**eighbors – kNN algorithm, where $k = 1$, 1-NN

PCA – **P**rincipal **c**omponent **a**nalysis – dimensionality reduction method transforming features into the space of the most informative eigenvectors

RF – **R**andom **f**orest – decision tree integration technique based on majority voting of base decision trees trained on different subsets of training dataset

SGD – **S**tochastic **g**radient **d**escent – an optimization algorithm for finding approximate extrema of differentiable functions

SVM – **S**upport **v**ector **m**achines – classification algorithm based on maximizng the margin between the classes

Introduction

Artificial intelligence and machine learning systems became a part of every field of our lives and are still gaining in popularity. It did not only take over the world of science, but also revolutionised many branches of industry. The methods of artificial intelligence are used, among others, in optimization problems, cybersecurity, healthcare, finance, law, education and media [17, 133].

An important role in artificial intelligence play the algorithms which are able to learn from the incoming, labelled data. After being fed with the data, they can make a decision or predict without being programmed to explicité. They form a field of artificial intelligence called machine learning [72].

A machine learning algorithm designed to be trained on known data in order to assign the unknown objects to a category is known as a classifier. Ensemble classifiers (classifiers built from multiple models) are of a particular interest to researchers and data scientists because of their classification quality [72].

Research problem

There are many possible approaches to classifier integration. There is not much research done yet in the field of geometric approach. It is based on the model representation in a coordinate system, where the axes are the equivalents of numeric features. The geometry–based reasoning can lead to interesting, unavailable for other types of ensemble methods results. Several works show the advantages of using geometric representation over other techniques, but they are not yet broadly used. One of the advantages is the better classification quality measured with one of many units. Because the research area is wide, the implications of using variations of the basic methods are not fully understood, especially in the field of decision trees [23, 102].

Research hypothesis. *Utilization of trained decision trees' decision boundaries allows for building an ensemble of classifiers with a greater value of performance quality measure than that of the random forest or majority voting using the same set of trained decision trees.*

Motivation and contribution

Geometric approach in classifier integration is not broadly spread in the context of decision trees. The main goal of these studies is to broaden the knowledge about the creation of decision tree ensembles by presenting novel algorithms and evaluating them in comparison to the existing and commonly used methods. In the experiments, majority voting and random forest are taken as the references, because of their popularity and effectiveness. Multiple classification quality measures are involved to account for the imbalanced datasets. For reliability, several open–source benchmarking datasets from UCI [38] and KEEL [1] platforms were used to evaluate the proposed methods.

Throughout the course of PhD studies, four different algorithms for classifier integration were proposed, implemented and evaluated using the datasets mentioned. Statistical tests conducted to compare the proposed with the referential techniques indicate significant improvement in several cases. Related works concerning SVM integration and purely theoretical considerations around classifier integration were published.

Dissertation layout

This chapter presents the research problem and motivation. The outline of the author's contribution is given.

In the chapter 1 the basics of multiclassifier systems are described. Common classification and ensemble methods are introduced and special consideration is given to the algorithms relevant to the presented works. The concept of local confidence is mentioned and several ensemble classifier application examples are given.

The chapter 2 introduces the reader to the idea of geometric representation in the context of classifier integration. The idea of decision boundary, Voronoi cells as well as static and dynamic space division is described.

The datasets used in evaluation, technologies used to implement the algorithms and perform data processing and results analysis are gathered in chapter 3.

The main content of the dissertation – presentation of the published results can be found in chapter 4. It is divided into four sections, each concerning a different algorithm.

The layout of all sections is similar – first the method's basics are described, then the results of the implementation evaluation are presented together with the statistical tests.

The chapter 5 concludes the dissertation. Additional experimental results can be found in the appendices A and B. They complement the publications from the first two sections with the results of the experiments involving a larger group of datasets and quality measures from the chapter 3.

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

Chapter 1

Ensemble learning

Machine learning is an important part of artificial intelligence. It can be divided into more specialized fields like supervised and unsupervised learning, reinforcement learning, deep learning, etc. This work concerns supervised learning, hence the following sections will deliberate on the topics relevant to this area.

1.1. Classification

Classification is one of the methods of machine learning, more specifically – supervised learning, where the correct labels of the objects are known a priori. The information about the labels is used in the prediction process to assign the unknown objects to one of the specified classes according to the similarity of this object to the members of each class [116].

Classification is a task of assigning a label (class) to an object described by its attributes. The label comes from a finite set of labels $\Omega = \omega_1, \omega_2, \ldots, \omega_C$. Each class contains similar objects, whereas objects of distinct classes are dissimilar. A binary classification task is a problem involving only 2 classes. A classifier is a specific algorithm performing the classification. It is denoted in the literature as $\Psi(x) \in \Omega$.

The attributes (features) are usually real numbers. They can be continuous (for example, blood pressure, voltage, probability), integers (height in centimeters, weight in kilograms, age in years) or categorical (sex encoded as numbers or Boolean values: false as 0 and true as 1). Values of categorical features do not matter for the classification – the choice is arbitrary and should be able to be changed without affecting the results. Hence, the classifier can be represented as a function mapping the space of real–valued features to a class that has the highest probability: $\mathbb{R}^N \rightarrow \Omega$ or as a set of discriminant functions $\{g_1, g_2, \ldots, g_K\}$ assigning a probability for every label i:

$$g_i : \mathbb{R}^N \rightarrow \mathbb{R},\ i \in 1, 2, \ldots, N$$

Ties are usually broken randomly, i.e. the one of the labels with the highest score is chosen [73].

There is a plethora of possible classification algorithms performing differently depending on the data. The right choice is often a hard, critical step in data analysis or its processing [17, 69, 73].

Classification belongs to a larger group of supervised learning techniques. They are based on training the classifier using labelled data (training dataset) and then predicting the class of unknown data (testing dataset) using the obtained model. Unlike supervised category, unsupervised learning operates on non–labelled data, for example, clustering is a process of grouping similar objects into pairwise distinct groups (clusters) knowing only the number of these groups beforehand [17, 73].

Yet another technique is reinforcement learning, where the algorithm operates on unlabelled data and learning occurs using the information on how well does the algorithm perform in the environment [69].

This section focuses on the methods essential for the dissertation. Other popular algorithms like Support Vector Machine or Neural Network will be introduced shortly for completeness.

1.1.1. Feature selection, dimensionality reduction

Every object has a set of features (attributes) which might be qualitative (ordinal, nominal) or quantitative (discrete, real–valued). If the features of the objects compose real–valued vectors $x = [x_1, x_2, \ldots, x_N]$, the $\mathbb{R}^N$ space they generate is called a feature space. The feature space is very often a subject of studies involving integration using geometrical functionals. Categorical data does not apply to this procedure because of the irrelevance of the features' values. Non–numeric values cannot be used with geometrical transformations [73].

In some cases, the number of features is overwhelming, which leads to high computational costs and noise. Some features can be naturally correlated with each other, which leads to unnecessary redundancy. In such a scenario, a procedure called feature selection is conducted to reduce the count of features taken into consideration [50]. One of the most common techniques is based on analysis of variance (ANOVA) and selects the most informative features (features with the greatest variance). Other techniques involve genetic algorithms, iterative models, etc. [73].

Another approach is to generate new features, not contained originally in the dataset. Such an example is principal component analysis (PCA), which generates a new feature space based on the eigenvectors of the original dataset. This technique, albeit old, is widely used and is a basis for many variations that keep emerging [5, 116].

1.1.2. Decision tree

Decision tree belongs to the simplest and most intuitive machine learning algorithms. It works by recursively partitioning the classification space [107]. Although it has been proposed more than three decades ago [103], decision tree and many its derivatives are very commonly used today [123]. Easy representation, low computational cost and high quality make decision tree one of the most powerful and popular approaches in data science [107].

The idea behind this technique is the partitioning of the input space into cuboid regions with edges aligned with the axes. The name is derived from the method of obtaining such a classifier. First, a feature that best divides the objects into two groups is chosen and the division is made. This split is the root of the decision tree. As a result, two regions are obtained. In the next step, another division is done in each of the regions. Generally, a tree of depth b can produce up to 2^b regions. The splits are done based on a metric different among the possible implementations, for example, information gain (used in CART implementation [18]) or gini impurity (used in ID3 [103] and C4.5 [104]). The procedure is conducted recursively until a satisfactory division by means of the metrics mentioned is obtained or a specific depth of the decision tree is reached [17, 108]. To classify an object, the region it falls into is determined by starting at the root node of the tree and following the path down to a specific leaf node according to the split criteria at each node.

The quality of the split is based on decision gain, which is the measure of how does the information entropy change after the split. Information entropy for a region r created by a split can be defined as follows:

$$E_r = -\sum_{c=1}^{C} p_c \log_2 p_c$$

where p_c is the probability of picking the class c, so $\sum_{c=1}^{C} p_c = 1$. It can be noticed that the entropy is lowest when for some c_0, $p_{c_0} = 1$ and $p_c = 0$, when $c \neq c_0$, then $E_r = 0$. When splitting a region, the entropies of regions are calculated before and after the possible split and weighted by the number of objects in each branch:

$$E = -\sum_{r=1}^{R} \frac{|D_r|}{|D|} \sum_{c=1}^{C} p_c \log_2 p_c$$

where $r = 1, 2, \ldots, R$ iterates over the leaves and D_r denotes the set of objects within the region r. The goal is to minimize the sum of weighted impurities over every region (maximize the information gain): $\mathrm{IG} = E_{\mathrm{before}} - E_{\mathrm{after}}$ [103].

Other popular metric is gini impurity. The mechanism is the same as described earlier, but instead of information entropy, gini imbalance is used instead:

$$\mathrm{GI} = \sum_{c=1}^{C} p_c(1 - p_c)$$

The lower the value of GI, the better the decision tree models the data [75]. The classifier is trained until a stop condition is met or the tree has the maximum depth.

1.1.3. Nearest neighbor

In the non–parametric classification, prototypes are used. A prototype is a representative element of a class. The classification process is based on the similarity of the object under test to one or more prototypes. Usually, similarity is defined using a geometrical definition, for example, the distance in a certain metric – the smaller the distance, the better the classification [73].

Such an example is the k–nearest neighbor (kNN) algorithm, where the prototypes are the training objects. Given a training dataset D with no identical objects with different class labels and some positive integer k, the kNN classifier finds for each object k nearest neighbors within D and assigns a label based on the plurality voting of the most similar prototypes.

A special case is 1NN, where the label of a tested object is determined by the single nearest training object. Graphically, the classification regions obtained by the 1NN rule can be depicted as Voronoi diagrams, where each region is a set of objects closest to a prototype x_0:

$$V(x_0) = \{x, x \in \mathbb{R}^N, d(x, x_0) = \min_{x_i \in D} d(x, x_i)\}$$

where $d(x_1, x_2)$ denotes the distance between the objects x_1 and x_2 in some metric. Euclidean metric is the most intuitive, but others, like Manhattan or Chebyshev's, can be less costly.

Usually, it is essential to find a smaller set of prototypes. Firstly, with a smaller number of prototypes, it is more efficient in terms of time and memory. It is important when dealing with big data. Secondly, in the process of refining the training dataset, some noisy objects from D could be filtered out, which could lead to the improvement of classification quality. To reduce the number of prototypes, either a subset of D is selected (prototype selection) or a set of new prototypes is created (prototype extraction). There are multiple

possible methods for finding subsets other than randomly like Hart's or Wilson's methods. Similarly, there is a plethora of extraction methods. The most intuitive one utilizes means (nearest centroid) to calculate one representative for every class. Other methods use neural networks (competitive learning), gradient descent, bootstrapping, tabu search and other heuristics [73].

1.1.4. Other supervised learning methods

There are many other techniques of supervised machine learning. Here, some popular methods are mentioned for completeness.

1.1.4.1. Support vector machine

One of the most popular classifiers is support vector machine (SVM). It is a decision machine – it does not provide posterior probabilities, but rather labels for the tested objects. In its most basic form – with a linear kernel, SVM solves the classification problem using a model of the form:

$$y(x) = w^T x - b$$

where x is the object under test, w is the normal vector to the hyperplane separating the classes and b is an offset [53].

Intuitively, SVM maximizes the margin between the two classes. Suppose there is a binary problem with labels encoded as $y_i \in \{-1, 1\}$, the feature space is normalized, so $x_i \in [0, 1]^N$. Let us also assume the classes are linearly separable, i.e., there exists a hyperplane that perfectly separates the training objects of the two classes. The hyperplane of course has $N - 1$ dimensions. If such a hyperplane exists, then there exists an infinite number of hyperplanes fulfilling this requirement. The notion of margin determines which one of these hyperplanes is the solution to the problem. Since we want to maximize the distance between the classification boundary and the training objects, we can write that $w^T x_i - b \geq 1$ when $y_i = 1$ and $w^T x_i - b \leq -1$ when $y_i = -1$. This can be simplified as

$$(w^T x_i - b) y_i \geq 1 \tag{1.1}$$

Now the problem is reduced to minimizing $||w||$ subject to 1.1. That is because the shortest distance between the classes and the margin occurs when $w^T x_i - b = 1$ for $y_i = 1$ and $w^T x_i - b = -1$ for $y_i = -1$ and is equal to $\frac{1}{||w||}$, so the margin between the classes is equal to $\frac{2}{||w||}$. This is called the hard margin. In some cases, this cannot be done analytically and numeric methods are used instead [53].

The situation where the objects of different classes are linearly separable is rare and the definition of the hard margin above does not work in this case. To solve this problem, the requirement in the equation 1.1 needs to be loosened and a penalty for wrong classification has to be introduced. Now the classification is parameterized with yet another value – penalty. The problem can be reduced to minimizing the loss function defined as follows:

$$\frac{1}{|D|}\sum_{i=1}^{|D|}\max(0, 1 - y_i(w^T x_i - b)) + \lambda||w||^2 \tag{1.2}$$

where λ is the parameter deciding on the tradeoff between the width of the margin and the number of misclassified objects. In the first term, hinge loss was introduced to penalize the misclassified objects. It vanishes when the object is out of range of the margin: $\max(0, 1 - y_i(w^T x_i - b)) \implies y_i(w^T x_i - b) \geq 1$. This means that even if the object is properly classified, however, it is in the range of the margin, some penalty is still applied [17, 53].

1.1.4.2. Artificial neural network

Artificial neural network (ANN) or shorter – neural network has its name derived from the biological neural networks composing brains of animals. The most basic building block of ANN are the artificial neurons – perceptrons inspired by the real neurons. They can take an input like the synapses of neurons and process the sum of inputs using some non–linear function to produce an output. The result of the processing can then be passed as an input to another neuron. This way, the neurons are composed of layers, each one responsible for a different kind of processing. Every neuron has a weight associated with it. Typically, the weights are assigned randomly at first and updated with each training iteration to increase or decrease the strength of the signal from the neurons. The first layer is called an *input layer*, the last – an *output layer* and the intermediate ones – *hidden layers*. ANN can have any number of hidden layers with any number of neurons in each layer, the limiting factor is the complexity of the model.

The training of a neural network is an iterative process. Firstly, an architecture of the ANN is designed (number of layers, number of neurons in each layer, connections between neurons, recursive flow and other additional elements). Then weights are initialized (usually randomly or uniformly). Then the labelled training data is split into batches and for each batch forward and backward propagation occurs.

During forward propagation, the output is calculated for every neuron, starting with the input layer using the formula:

$$y = f\left(\sum_{i=1}^{I} x_i w_i + b\right) \tag{1.3}$$

where x_i denotes the input from i-th neuron, w_i – weight assigned to i-th neuron, b – a bias and f is an activation function. Activation function is a non–linear function (otherwise ANN would be a linear classifier or even a set of linear equations) turning an unbounded input into a value of the unit range: $f : \mathbb{R}^I \rightarrow [0, 1]$. One of the most commonly used activation functions is the sigmoid function: $\sigma(x) = \frac{e^x}{1+e^x}$. Other useful functions include:

- relu – returns the input only if it is not negative: $f(x) = \max(0, x)$, it discards all the negative input,
- leaky relu – similar to relu, but allows negative values to "leak" after scaling by some factor $\alpha \in (0, 1)$: $f(x) = max(x, \alpha x)$,
- softmax – turns the input into a result with the probability distribution, hence it is usually used as the activation function of the output layer; the input is turned into an exponential function and the outputs are normalized.

Other activation functions include exponential, hyperbolic tangent, selu, elu. Custom functions can also be used. Using the output of the entire ANN and the encoded label, the value of a loss function L is calculated.

Having the value of the loss function, an update of the weights of the ANN is conducted in the process of backpropagation. Usually, the stochastic gradient descent (SGD) algorithm is used to calculate the new values for weights:

$$w_i \leftarrow w_i - \eta \frac{\partial L}{\partial w_i} \tag{1.4}$$

where $\frac{\partial L}{\partial w_i}$ is the derivative of the loss function with respect to the weight w_i and η is the learning rate. The higher the learning rate, the more aggressively are the weights updated. The same procedure applies to the bias values. There are more optimizers that can be used:

- adagrad – an optimizer with specific learning rates; the adaptation relates to how frequently is the parameter updated. The more updates a parameter receives, the smaller the updates,
- adam – an efficient optimization based on stochastic gradient descent that uses adaptive estimation of the first–order and second–order moments to accommodate to the sudden changes.

Other popular algorithms include adamax, adadelta, nadam, ftrl.

The algorithm 1 presents the process of training the ANN.

Algorithm 1: ANN training.

Input: Training data D, ANN architecture
Output: Trained ANN
1 Initialize the weights of the ANN.
2 **while** *next batch* B_i *of* D *exists* **do**
3 Calculate the output of the ANN in forward propagation for the batch B_i using 1.3.
4 Update the weights of the ANN in backward propagation using 1.4.
5 **end**

Specialized types of ANN have emerged to solve specific tasks, for example:

- convolutional neural networks – composed of one or more convolutional layers, uses tied weights and pooling layers, performs image filtering; superior in image and speech recognition,
- recurrent neural networks – propagate data forward and backward – from later processing stages to earlier stages, hence it has a memory; powerful in natural language processing.

1.2. Combining classifiers

Each pattern is characterized by the feature vector x. The recognition algorithm Ψ maps the feature space x to the set of class labels Ω according to the general formula:

$$\Psi(x) \in \Omega. \tag{1.5}$$

Let us assume that $k \in \{1, 2, ..., K\}$ different classifiers $\Psi_1, \Psi_2, \ldots, \Psi_K$ are available to solve the classification task. In multiclassifier systems (MCS) these classifiers are called base classifiers. In the binary classification task, K is often assumed to be an odd number to avoid ties. Otherwise, the definition must be enhanced with random draws or weighting. As a result of all the classifiers' actions, their K responses are obtained. Usually, all K base classifiers are applied to make the final decision of the MCS. However, some methods select a subset of base classifiers from the ensemble, in particular cases only a single one can be selected. The main goal of creating classifier ensembles is to provide a better classification quality than of the base models [86].

The combining method is applied to make the final decision of the ensemble of classifiers (EoC) based on the outputs obtained. One of the most commonly used classification algorithm–agnostic integration technique is the majority voting [73]. Other techniques include average, minimum, maximum, median, oracle [68, 72, 89]. These abstract techniques can easily be applied to any type of classification algorithms. Probably the most widely used integration technique for decision trees as base models is the random forest [20].

1.2.1. Ensemble creation phases

In general, the procedure of creating the EoC can be divided into three major, consecutive steps [73]:

1. Generation – a phase where the classifiers are trained and the pool of base classifiers is created.
2. Selection – an optional phase where only several models from the committee are taken to the next phase.
3. Integration – a process of combining the outputs of multiple classifiers to obtain a single, integrated classification model. This step is optional if the selection phase results in a single classification model.

In the generation phase, each base model can be trained using the injection of randomness into the training set or partitioning of the dataset [109]. In horizontal partitioning, the original dataset is divided into several sets that include all features, while in vertical partitioning each base model uses a subset of all features [29]. With this procedure, different base models are obtained. Another way to create diversity in the base models is to train them with varied parameter values.

During the selection phase, the competence of each base classifier can be used [21]. The simplest way to do this is by measuring the classification quality for each model. The worst classifiers can be then omitted in the integration phase and the best classifiers can have a greater impact on the integration process. However, it was noticed that using local competence provides better results compared to generalizing over the whole feature space, since the models can perform differently depending on the classification area. The problem of diversity of base classifiers occurs often. One of the methods to solve it is by producing a pool of classifiers followed by the selection to choose the models that are most diverse. Some of the techniques include:

- Using the diversity matrix of base classifiers to select the least related iteratively until the desired number is reached [46].

- Clustering the base models using the diversity matrix and picking one model from every cluster. This procedure is based on the assumption that models from different clusters make errors in diverse regions [45].
- Using kappa–error plots to remove the subset of the base classifiers [82].

The integration phase can be performed using different types of classifier output [73, 101]:

- a class label,
- a subset of labels ordered by plausibility,
- a vector of all possible labels with the corresponding support values.

One of the most used methods to integrate the class labels of base classifiers is the majority vote rule (MV). In this method, each base model has the same impact on the final decision of EoC. In the weighted majority voting rule, the integration phase includes probability estimators or other factors of the base models to the final decision of MCS [22, 81].

Simple non–trainable combiners can be used to integrate the output of the base classifiers in the form of vectors of probabilities of each class for a given tested object. Let us denote such a vector for k-th classifier as $[\Psi_{k,1}(x), \Psi_{k,2}(x), \ldots, \Psi_{k,C}(x)]$, where x is the object under test. Then the support value of the integrated classifier can be written as $\mu_k(x) = f(\Psi_{k,1}(x), \Psi_{k,2}(x), \ldots, \Psi_{k,C}(x))$, where f is a combination function. The combination function can be given, for example:

- average: $\mu_k(x) = \frac{1}{C} \sum_{c=1}^{C} \Psi_{k,c}(x)$,
- maximum: $\mu_k(x) = \max(\Psi_{k,1}(x), \Psi_{k,2}(x), \ldots, \Psi_{k,C}(x))$,
- product: $\mu_k(x) = \prod_{c=1}^{C} \Psi_{k,c}(x)$.

Generalized means can be used in place of the combiners mentioned or even custom functions.

Another approach to compose the vectors of support functions is to use trainable combiners. This family of techniques is based on an additional validation phase. An example is the weighted mean, where the weights for base classifiers are derived from the error estimations [73].

Decision templates are class–independent. The decision template combiner remembers the most specific decision profile for every class, called the decision template, and then compares it with the decision profile of the object under test using some similarity measure. The closest match is returned as the label of the tested object [73].

The selection phase can be omitted and then all models take part in the integration process. Xu et. al developed several methods in this manner. Every classifier is trained on the entire space, which means that the base models are competitive rather than complementary. The results on the recognition of totally unconstrained handwritten numbers indicate that the performance of individual classifiers could be improved significantly by the combination approach proposed [134].

1.2.2. Majority voting

The majority voting is a combining method that works at the abstract level. This voting method allows for counting the base classifiers' outputs as a vote for a class and assigns the input pattern to the class with the greatest number of votes. There are 3 common consensus patterns:

- unanimity – all decision makers agree,
- a simple majority – the decision must receive more than 50% of the votes,
- plurality – the decision that receives most of the votes wins.

Plurality is the most widely spread approach in classification, because it avoids the situations when no decision can be made. It will be referred to by majority voting further on.

The majority voting algorithm can be written formally as follows [5]:

$$\Psi_{MV}(x) = \arg\max_{\omega} \sum_{k=1}^{K} I(\Psi_k(x), \omega), \tag{1.6}$$

where $I(\cdot)$ is the indicator function with the value 1 in the case of the correct classification of the object described by the feature vector x, i.e. when $\Psi_k(x) = \omega$. In the majority vote method, each of the individual classifiers takes an equal part in building EoC unless they are removed in the process of classifier selection, the competence of the models is irrelevant [5, 73].

The variation of MV technique is weighted majority voting (weighted MV). The classifiers are trained on the entire classification space similarly to the base method. Afterwards, their quality is evaluated using the validation subset and weights are calculated using the resulting quality measure values [55]. This procedure has proven itself to yield better results in some cases than unweighted MV, where all classifiers are treated equally [90].

1.2.3. Random forest

One of the most popular ensemble methods is a Random Forest (RF) introduced by Breiman in 2001 [20, 110]. This technique has proven itself to be very powerful and

many related algorithms appeared over the years. In the article [40], 179 different classification algorithms were evaluated using 121 datasets. The results show that random forest outperforms the majority of the examined classifiers. It operates on a pool of base classifiers – decision trees trained on different subsets of the training dataset. The object under test is classified by every model and the results are gathered. Finally, majority voting is applied to obtain the most common label.

The random forest is just an integrated classifier consisting of a set of tree–structured classifiers, each tree grown with respect to some random vector. The differentiation between the base models can be introduced by randomly sampling from the feature set, from the dataset, or just varying some of the parameters of the classifier. Any combination of the sources of diversity mentioned will generate a random forest.

Similarly as in simple MV, weights can be introduced in RF. This topic was investigated by Akash et al. [4]. The weight calculated from the local confidence was assigned to each leaf of every decision tree in the pool of the random forest. 25 benchmarking datasets were used to prove the improvement in classification quality over the base method. Similar approach was examined by Gwetu et al. [51]. They introduced two different techniques. The first one is an extension of the idea of the leaf node purity. The rate of purity convergence and the depth it occurs are taken into consideration. The second one takes into account the confidence with which a tree makes both correct and incorrect classifications in a comprehensive manner.

Extreme Gradient Boosting (XGB) is among the most widely spread algorithms, especially in machine learning competitions [33, 117, 121]. The algorithm works by training subsequent decision trees, where consecutive models minimize the value of a loss function generated by its predecessor [44]. Another implementation of Gradient Boosting Decision Tree aiming at performance, especially in the case of high dimensionality, is LightGBM [64]. Without loss of performance in classification, the process of training a model can be sped up up to 20 times.

1.3. Local confidence

Multiple works indicate that treating all classifiers trained on the entire classification space equally does not necessarily yield the best results [31, 70]. This approach implies training classifiers on data from some region or qualifying the model in this area. The metainformation is used to compose the final classifier [124].

In the paper [6] two implementations of linear competence are investigated: cooperative and competitive. In the former, a weighted mean of the outputs of the local perceptrons is calculated with the weight being a function of the distance between the

input and the perceptron's position. In the latter model, the cost function implies only one of the local perceptrons to be the final output. The results show that the proposed method generalizes much better than the referential multilayered perceptrons and uses much less memory.

An interesting approach is presented in [36], where Ding et al. introduce local–global classifier integration. They argue that the local classifier's focus is to detect a subtle presentation of the disease, leveraging information in radiology reports that roughly indicates the location of the abnormalities. On the other hand, the global classifier represents the dominant spatial structure of the image. Finally, the two complementary models are combined using weighted linear fusion, where the weight of each output is computed from the confidence probabilities of the two classifiers. The method was evaluated on 3 datasets, which demonstrated the improvement of the classification quality of the local–global fusion method over any base model.

The idea of local confidence was combined with a technique similar to that presented by Gwetu et al. as mentioned in 1.2.3. Armano and Tamponi have developed a forest of local trees [8]. The main concept is to train the base decision trees on the subsets of the training dataset contained in a specific region. The base models become experts in possibly overlapping regions that compose the entire classification space. Comparative experimental results have shown that this method performs better than other referencial ensemble classifiers.

Advantage over common methods like random forest, majority voting, AdaBoost and diversity regularized ensemble pruning was observed by Cai in [28]. The procedure consists of two steps. In the first one, base classifiers are trained and pruned. In the second stage, the pruned models are weighted locally using a fusion method, which is the nearest neighbor of the testing data points.

1.4. Applications

Classifier ensembles are used in almost every field of science and industry.

1.4.1. Finance

MCSs are broadly used in financial data analysis and provide aid in decision making. They are effectively used by banks in property valuation, bankruptcy prediction and credit scoring [3, 49, 77, 128] but also by manufacturers and sellers in demand forecasting – the process of constructing models to predict the quantities of products that customers will purchase [59].

The other area of application is fraud detection. Multiple works on credit card fraud recognition use real–life data of transactions from an international operation using credit cards. A comparison of RF, logistic regression and SVM was conducted to determine the best model in terms of classification quality. The best results were obtained with the use of RF [11, 133].

Trade based stock market tries to manipulate the stock values by causing artificial traffic. An interesting research track involves the usage of a peer–group analysis for trade stock manipulation recognition, based on the detection of outstanding values whose dynamic behavior differs from that of previously similar stock values [67].

1.4.2. Image analysis

Ensembles of classifiers have found their use in image analysis and classification [71]. They are used in geography in landslide susceptibility and spatial prediction of landslides [95, 96]. Manufacturers employ ensembles to detect defects in their products [94]. Many applications in electronics are associated with image classification, for example, defect detection in sensors and semiconductor manufacturing [112, 131]. Convolutional networks are often enhanced with integration techniques to improve the classification quality [63].

Many applications in image analysis are tightly related to medical imaging. Ensembles have been applied to the classification of fMRI [97] or structural MRI [125] data.

1.4.3. Medicine and biology

One of the best examples of classifier integration usages is in medical data processing. They are used in diagnostics of headaches, diabetes, epileptic seizures, cancers [7, 10, 60, 114] and analysis of sensor data like detecting abnormal heart sounds [41, 100]. Medical imaging is very often enhanced with classification. This helps in finding anomalies and differentiating between pathologies, for example, between similar cancer types [37] but also on the molecular level with DNA and RNA defect detection [30, 56, 78]. They can be helpful in chemical reactions prediction, for example protein–protein interactions and elucidating the molecular mechanisms underlying protein [2, 76, 130]. Metagenomic classifiers have the ability to identify taxa at the genus, species, strain levels, quantify the relative abundances of taxa [83].

Computer Aided Diagnosis (CAD) is based on artificial intelligence. Heterogenous set of classifiers composed of SVM, Bayesian networks and ANN was found to be effective in cardiovascular disease diagnosis [39, 133]. Many CAD systems designed to

diagnose coronary diseases are based on the electrocardiogram. It helps in early prediction of coronary artery disease [34].

1.4.4. Security

Multiple classifiers are employed in face and voice recognition [41]. Masquerade detection is another field where MCSs are used. Masquerade detection system determines whether a given computer's activity corresponds to a target user's, thereby detecting whether a masquerader has stolen the computer session of an owner [85, 111]. For any given cyber–attack, multiple methods of detection have been developed [88, 122]. Results have been reported that MCS outperform other approaches in active learning needed to update the models and keep up with the dynamic changes in the malicious code versions [115].

Chapter 2

Geometric approach to classifier integration

Considerations about classifier integration using their geometrical representation have been studied for over a decade now [102]. Based on operations in geometrical space generated by real-valued features, this procedure has proven itself to be effective in comparison to others, commonly used integration techniques such as majority voting [23]. Geometric representation enables to leverage additional properties of data even in natural language processing [58].

2.1. Decision boundary

Several classifier integration approaches based on the geometric representation of the decision boundary between classes have emerged. It has been noticed that calculating local confidence based on the distance between the object under test and the decision boundary can lead to an improvement in classification performance. Trajdos and Burduk described a method for linear model integration by deriving a scoring function from the distance to the classification boundary [127]. The probability functions proposed take into account not only the distance between the recognized object and the decision boundary but also the prior probability of the class labels. The effectiveness of this method is demonstrated on 70 open–source benchmarking datasets.

Another work shows two possibilities to calculate the scoring function from the distance to the classification boundary [126]. Together with 4 different combination methods, the algorithm was tested and statistical analysis was provided. The result proved that the combination strategies based on simple average and trimmed average perform the best in the context of geometrical combination.

Burduk and Kasprzak investigated the usage of geometric mean in thecalculation of the decision boundaries of the integrated classifier [27]. The experiments were conducted in two-dimensional feature space for binary classification tasks. 3 base classifiers were integrated. According to Friedman ranked test, the proposed integration algorithm is better than MV method.

In one of the previous papers, the author together with Burduk have shown significant improvement in the classification by applying weighted mean and median functional to the decision boundaries of the Support Vector Machine (SVM) classifiers [24]. The paper can be thought of as a continuation of the studies from the previous one by introducing other functionals and different counts of base classifiers. Similarly, the results of the experiments and statistical analysis are appealing.

Ksieniewicz and Burduk have enriched this idea by training on clustered data [71]. In their proposal, clustering is first conducted, so that the local expert models can be trained. Afterwards, the weighted scoring function based on the distance to the classification boundaries is calculated and utilized in the integration phase. The effectiveness of this method is proven by extensive experiments on benchmarking data and statistical analysis.

Distance from the decision boundary can be used to improve not only the integration methods as presented in [26]. In this paper, a modification of a gentle AdaBoost technique is presented, leveraging the distance of the objects from the decision boundary.

Since many powerful methods employing decision boundaries have emerged, effective techniques for extraction have been developed. Gong proposes an algorithm, which calculates the boundary not by inspecting the classification mechanism, but rather the output of the classifier [47]. The experimental results show that the boundary converges fast.

2.2. Voronoi cells

A geometric approach to the classification problem by Voronoi cells utilization was recently examined by Polianskii and Pokorny [98]. The most common approach utilizes point–wise cell membership information by means of nearest neighbor lookup and does not use other geometric information contained in Voronoi cells because the computation of the Voronoi diagrams is computationally expensive. The authors consider Voronoi cells instead of points as the basic objects being classified and propose a Monte Carlo integration–based technique that calculates a weighted integral over the boundaries of the Voronoi cells. Boundaries of the cells are associated with labels of the closest training

objects. Then, integration over the boundaries regarding the associated labels is performed to obtain the most probable class. This approach was tested using SVM, nearest neighbor and random forest classifiers. The experiments indicate that the method presented performs similarly to Random Forests for large dataset sizes, but the classification quality for smaller datasets increases significantly.

Nearest neighbor classifiers are proven to be efficient when testing which Voronoi cell an object belongs to [12], because there is no need to calculate the geometry of every Voronoi cell. An efficient search lookup was proposed by Kushilevitz et al. [74]. Originally, the algorithm was employed to improve the performance of computation in NN classifiers. The algorithm employs a space–efficient data type that allows to approximate the nearest neighbor in time nearly quadratic regarding dimensionality. This procedure was found to be especially efficient for imbalanced datasets [118].

However, the nearest neighbor algorithms are difficult in usage regarding specifying the number of prototypes. Using too many leads to high computational complexity. Too few prototypes can cause an oversimplified representation of the decision space, especially for datasets that are not linearly separable, have island–shaped decision space, etc. Multiple solutions for this problem were proposed. One way to achieve this is to apply Generalized Condensed Nearest Neighbor rule to obtain a set of prototypes [65]. Each prototype is an object of the training dataset. The novelty of this method in comparison to the previous ones is that instead of combining methods based on decision templates by employing a single prototype for each class, a prototype selection method to obtain a set of local decision prototypes is used. It represents the decision space better by avoiding drawbacks caused by insufficient number of training samples, island-shaped decision space distribution, and classes with highly overlapped decision spaces. To determine the class of an object, its decision profile is computed and then compared with other decision prototypes.

Another approach was proposed by Gou et al.: a kNN-based classifier inspired by the local mean–based k–nearest neighbor and pseudo–nearest neighbor rules [48]. The first step of the algorithm is to apply the kNN algorithm to obtain a fixed number of prototypes for every class. Then the local mean vectors are calculated to transform the set of prototypes to better represent the decision space distribution. The proposed algorithm is very promising as the results of the experiments on 39 UCI datasets suggest.

Nguyen et al. proposed a method based on the distance from the metaclassifier [91]. A granular prototype for every class from the meta-data from observations from the training dataset is constructed with the same class label. Every such prototype can be represented as a vector of intervals, where the fuzziness reflects the uncertainty of the class

prediction of the base models. The class label during tests is predicted by choosing the class label of the granular prototype that is the closest to the object under test. Experiments were conducted against AdaBoost, Random Forest, Decision Template, TSES, Decision Tree C4.5, and L2LSVM using 26 datasets. Statistically significant improvement of performance of the proposed method was shown.

2.3. Combining classifiers using space division

The idea of combining classifiers using a structure, so that the competence of each classifier is not constant across the entire classification space, is not new. There are two major types of approaches: static and dynamic space division. In the static division, the structure is known a priori and is unrelated to the input data, whereas in dynamic technique, the data used determines how is the division conducted [54].

2.3.1. Static space division

The static division technique dates earlier than the dynamic one. Woods et al. notice that, after training base classifiers, selecting the expert models from the region surrounding the test object results in a better classification performance in comparison to the approach where all the classifiers from the pool are treated equally. This technique was compared with other methods using empirical studies using 5 datasets and the improvement was confirmed [132].

The author of this dissertation has conducted experiments in the integration of SVM classifiers. Binary classification tasks are considered. Geometrical representation as a linear function of SVM models with linear kernel was used. The regions of competence are set before the training by splitting the classification space along one feature axis into equisized regions. Then the models are trained and the median functional is applied to the selected classifiers within each region. This means that the selection procedure is conducted independently for every region. Since the selection might result in picking different experts in each area, the integrated classifier, being median of the remaining models, is not necessarily a linear classifier. This procedure has proven itself to improve the classification results compared to the referential MV [25].

The topic was studied deeper by the author in [24]. Additional experiments were conducted to compare the results obtained when using the weighted mean. Significant improvements were observed in comparison to MV as indicated by statistical tests.

2.3.2. Dynamic space division

Dynamic approach utilizes the input data to derive the structure used to ensemble the classification models. This can be done using a simple transformation of the output from the training classifiers or by employing another learning model [61].

It has been noticed that the local quality for each of the base classifiers might differ. The objective of classifier selection is to choose one or a subset of possible base classifiers to perform classification over a region. If the division is known a priori, the selection is called static. Otherwise, the models are tested for their quality for the new pattern [99]. Kim and Ko [66] favor local confidence over averaging the quality of classification over the entire space. The method qualifies the local confidence of each model on the training dataset and during testing the confidence is used to evaluate the outputs of base classifiers. Combining the complementary characteristics of the base models is proven to outperform individual classifiers and several other integration methods.

Garcia et al. present an ensemble classification technique using feature space partitioning [80]. The main goal of the paper is to improve the classification quality of the imbalanced datasets. A hybrid metaheuristic was utilized to optimize the parameters related to the partitioning of the feature space.

An interesting approach is combining weighting with local confidence [120]. Their target is to improve the quality of the Facial Expression Recognition. The authors of the mentioned article notice that a classifier trained on a subset of training data should be limited to the area it spans with an impact on the resulting classifier. A base model should be assigned a higher weight in the regions near to their training space and a lower weight in the regions far from them - a Dynamic Weight Majority Voting mechanism for base classifiers is introduced. The experimentation results indicate that the proposed approach has the highest generalization ability.

The problem of generalization of majority voting was studied in [52]. The authors are using a probability estimate calculated as the percentage of properly classified validation objects over geometric constraints. Separately are considered regions that are functionally independent. A significant improvement in the classification quality was observed when using the proposed algorithm, although knowledge of the domain is needed to provide a proper division. The authors are using a retinal image and classifying over anatomic regions.

Another variation on weighted majority voting is the class–wise majority voting covered in [105], where a substantial number of ensemble strategies have been explored. Weights are determined for each label separately over the entire validation dataset. This can lead to the improvement of the performance of the resulting integrated classifier.

2.4. Diversity between classifiers

Having different base models is crucial in ensemble techniques. The most common methods of diversifying the base classifiers are bagging and boosting. Bagging is a method for generating multiple versions of a model and combining them to get an aggregated predictor. During the aggregation, an average over all the versions is taken when predicting a numerical outcome. Multiple versions are created with bootstrapped replicas of the training set [19]. On that basis more sophisticated methods for dataset partitioning have emerged [9].

Boosting, albeit not theoretically constrained, imposes a learner improvement by iterative training of weak learners and adding them to the final model. In this approach, the training data is weighted. Upon adding, the data is reweighted to stress the importance of the objects classified incorrectly, so that there is a higher probability that the next generation of the classifier will avoid previous mistakes [113]. The first successful boosting algorithm was AdaBoost, for which the authors received a Gödel Prize in 2003 [42, 43]. The algorithm of AdaBoost is presented as the algorithm 2 [17]:

Algorithm 2: AdaBoost algorithm.

Input: Number of iterations J, object under test x
Output: Label of the tested object

1 Let D be the cardinality of the training dataset. Initialize the weights for every object equally: $w_i^{(1)} = \frac{1}{D},\ i = 1, 2, \ldots, D$.
2 **for** $j \leftarrow 1$ **to** J **do**
3 Fit the classifier Ψ_j by minimizing the error function.
4 Evaluate the quantity $\epsilon_j = \frac{\sum_{i=1}^{D} w_i^{(j)}(1-I(\Psi_j(x_i),\omega_i))}{\sum_{i=1}^{D} w_i^{(j)}}$ where ω_i is the true label of an object described by the attribute vector x_i.
5 Calculate the coefficient $\alpha_j = \ln \frac{1-\epsilon_j}{\epsilon_j}$.
6 Update the weights: $w_i^{(j+1)} = w_i^{(j)} \exp\left(\alpha_j(1 - I(\Psi_j(x_i), \omega_i))\right)$
7 **end**
8 Make prediction using the final model $\Psi_I = \operatorname{sign}\left(\sum_{j=1}^{J} \alpha_j \Psi_j(x)\right)$

The diversity between base classifiers can also be obtained by using vertical or horizontal partitioning [107]. It has been proven that for datasets of extreme size (very large or small), horizontal partitioning (splitting data into disjoint subsets) outperforms other ensemble methods like bagging or boosting [32]. This provides great possibilities in parallelizing model learning in a distributed environment like p2p network [79].

Chapter 3

Experimental setup

This chapter presents details about the implementation of the algorithms and experiments. It describes the technologies and datasets used, metrics calculated and statistical tests conducted in order to evaluate the proposed methods.

3.1. Datasets

The experiments were conducted using the open–source datasets available on the platforms UCI Machine Learning Repository [38] and KEEL Data Set Repository [1]. Datasets are presented in the table 3.1. The imbalance ratio Imb is shown to stress the need to use metrics sensitive for highly imbalanced datasets (MCC). For all datasets, the feature selection process [50, 106] was performed to indicate the two most informative features.

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): $\mathrm{Imb} = \frac{\#\text{major class objects}}{\#\text{minor class objects}}$ [93]. If the value of Imb equals 1, then the dataset is balanced – all classes have the same number 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 metrics other than average accuracy should be considered when comparing the performance of the classifiers. The reason is explained in the following example. Suppose $\mathrm{Imb} = 9$ for a binary classification problem. When a classifier labels all the test objects with the label of the major class, its accuracy is $\mathrm{ACC} = \frac{9}{9+1} = 90\%$. In the parentheses, together with the names, abbreviations of the datasets' names were placed by which they will be further referred to for brevity.

Table 3.1: Description of datasets used in the experiments (name with abbreviation, number of instances, number of features, imbalance ratio).

Dataset	#inst	#f	Imb
Absenteeism at work (aa)	740	21	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)	19020	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

3.2. Technologies

Decision tree implementation from Spark was utilised [87]. In the 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 homogenity measure. Continuous feature discretization is conducted using 32 bins. The library was written in Scala and provides the application programming interface (API) in other popular languages like Python, Java, R. In the experiments, the native Scala API was used. The statistical tests were conducted in numpy [92], scipy [62] and pandas [84] using the native Python API. Figures were plotted using matplotlib [57]. The code used for the implementation of the presented algorithms is hosted on github, the links will be shared together with each algorithm description. The project for results analysis and conducting statistical tests is also hosted on github: https://github.com/TAndronicus/classifier-integration-analysis.

3.3. Methodology

The experiments were conducted 10 times for every algorithm and hyperparameter setup. The obtained results were then averaged to provide reproducible results. In all the presented algorithms, diverse versions were evaluated together with referential ensemble techniques: MV and RF.

Let us denote the cells of confusion matrix for a binary classification problem as TP – true positive, FP – false positive, FN – false negative, TN – true negative. Generally, for multiclass classification, a confusion matrix can be derived for every class ω_c, where $c \in \{1, 2, \ldots, C\}$: TP_c – true positive, FP_c – false positive, FN_c – false negative, TN_c – true negative. To compare the proposed ensembles with the existing ones, several metrics were used:

- $\mathrm{Accuracy}$ (ACC) – the ratio of properly classified objects, $ACC = \frac{TP+TN}{TP+FP+FN+TN}$,
- $\mathrm{Matthews\ Correlation\ Coefficient}$ (MCC) – more balanced measure than ACC, $MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FP)}}$,
- $\mathrm{Precision}$ – class agreement with positive labels, $\mathrm{Precision} = \frac{TP}{TP+FP}$,
- Recall – effectiveness to recognize positive labels, $\mathrm{Recall} = \frac{TP}{TP+FN}$,
- $\mathrm{F-score}$ – harmonic mean of precision and recall, $\mathrm{F-score} = \frac{2TP}{2TP+FP+FN}$,
- $\mathrm{Precision}_\mu$ – microaveraged precision, $\mathrm{Precision}_\mu = \frac{\sum_{c=1}^{C} TP_c}{\sum_{c=1}^{C} TP_c+FP_c}$,
- Recall_μ – microaveraged recall, $\mathrm{Recall}_\mu = \frac{\sum_{c=1}^{C} TP_c}{\sum_{c=1}^{C} TP_c+FN_c}$,
- $\mathrm{F-score}_\mu$ – microaveraged F-score, $\mathrm{F-score}_\mu = \frac{2\ \mathrm{Precision}_\mu \mathrm{Recall}_\mu}{\mathrm{Precision}_\mu+\mathrm{Recall}_\mu}$,

- $\mathrm{Precision}_M$ – macroaveraged precision, $\mathrm{Precision}_M = \frac{1}{C}\sum_{c=1}^{C}\frac{TP_c}{TP_c+FP_c}$,
- Recall_M – macroaveraged recall, $\mathrm{Recall}_M = \frac{1}{C}\sum_{c=1}^{C}\frac{TP_c}{TP_c+FN_c}$,
- $\mathrm{F-score}_M$ – macroaveraged F-score, $\mathrm{F-score}_M = \frac{2\,\mathrm{Precision}_M\mathrm{Recall}_M}{\mathrm{Precision}_M+\mathrm{Recall}_M}$.

When the experiments are conducted on multiclass datasets, as the classification evaluation metrics micro– and macro–average precision, recall and F-score are used. F-score is the harmonic mean of precision and recall. For this reason F-score takes both false positives and false negatives into account. Additionally, the overall accuracy is always presented. The F-score was computed alongside the accuracy because of the high imbalance of the multiple datasets used, as it was indicated in the section 3.1 for MCC. 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. The accuracy can be in this case artificially high. The metrics are calculated as defined in [119].

Statistical tests were performed to compare the results obtained for the proposed algorithms and references. According to Demsar et al. [35] the most reliable are non–parametric and post–hoc tests. In the presented algorithms, Friedman rank tests were conducted. Based on the ranks, post–hoc tests were pursued: Bonferroni–Dunn or Nemenyi, depending on the number of referential methods. The critical values were calculated for the confidence level of $\alpha = 0.1$.

Chapter 4

Proposed algorithms

This section presents the work done in the area of decision tree integration in geometric space. Every section treats about a different algorithm that was published or is in the process of publishing. The results in the first two papers do not cover all of the datasets presented in section 3.1. Extensive experiments on all possible datasets were conducted for completeness.

4.1. Integration of decision trees using distance to the centroid and to the decision boundary

The paper [15], concerning integration of decision trees, introduced a novel method, where the distance of the object from the decision boundary and the center of mass of the objects belonging to one class are used simultaneously in order to determine the scoring function. This means that the weights assigned to the class label by each classifier depend on the distance of the classified object from the centroid and from the decision boundary. The proposed technique was evaluated using open–source benchmarking datasets. The results indicated an improvement in the classification quality when compared with the referential ensembling algorithm – random forest.

When considering a two–dimensional space problem, there is a clear–cut representation of the decision tree as a finite set of rectangular, disjunctive areas with a specified label building up the complete decision space. In a multidimensional case, the areas are hypercubes. The article describes a method for obtaining an ensemble classifier inspired by the nearest centroid algorithm and the geometrical representation described.

4.1.1. Proposed method

The proposed algorithm introduces weighting to the majority voting, resulting in a classifier that assigns the weights to the classes for every tested object. The distances from the classification boundary and from the centroids are used to compute the final weight value. The goal of the article was to compare the described integration algorithm with the referential ensembling method (random forest). Two well–known quality measures (MCC and ACC) were calculated and used in the statistical tests. Additionally, the influence of the mentioned distances on the performance of the classification was studied.

As mentioned earlier, two values are calculated during the testing phase for every object: the distance from the centroid and the distance from the nearest decision boundary. The distance from the decision boundary is defined as the minimal distance from an object to the point that would be assigned a different label by the decision tree. This can be formalized using the following equation:

$$dist_B(x_0, \Psi) = \min_{x \in X; \Psi(x) \neq \Psi(x_0)} (dist(x; x_0)),$$

where X denotes the classification space, i.e. the cube generated by the possible linear combinations of vectors in the feature space. The formula is more general in the sense that it works for any classification algorithm. When the representation of the trained model is difficult to describe in terms of analytical functions, the decision space must be scanned in search for the solution. The case of decision tree is straightforward – for any given x_0, the closest point along the feature axes needs to be found.

By the distance from the centroid, the centroid labelled with the same class as the object is meant. The coordinates of the centroids for all labels are computed using the training subset.

The following equation formalizes the given distance:

$$f(dist, \beta, \gamma) = \exp(-\gamma(dist - \beta)^2). \tag{4.1}$$

The first step is the feature normalization. This reduces the bias associated with the calculated distances. In the paper, the β parameter (the mapping function 4.1) for the distance from the centroid of the class label – ω is fixed as $\beta_\omega = 0$ and for the distance from the decision boundary as $\beta_B = 0.5$. With that parameter set, the mapping function reaches its maximum at 0 and minimum at 1 when mapping the distance from the centroid and maximum at 0.5 and minima at 0 and 1 when mapping the distance from the decision boundary. This implies, intuitively, that the objects too far or too close to the decision boundary are scored with the smaller weight and, conversely, the closer the object to the centroid, the larger the weight associated with it. The γ parameter combinations for the respective cases (γ_ω and γ_B) are depicted in table 4.1. The combination

$\gamma_B = 20$ and $\gamma_\omega = 5$ is the base case, where the value of the function for the range of $[0, 1]$ varies between 0.0067 and 1. This way the minimal value is less than 1%. Additionally, cases where $\gamma_B = \gamma_\omega$ were studied. The α parameter is the share of the distance from the decision boundary in weight computation. In the experiments, α is the variable. The mapping functions are visualized in the figure 4.1.1.

Table 4.1: Combinations of γ parameter of the mapping function (4.1) examined.

γ_B	γ_ω
20	5
5	5
20	20
10	10

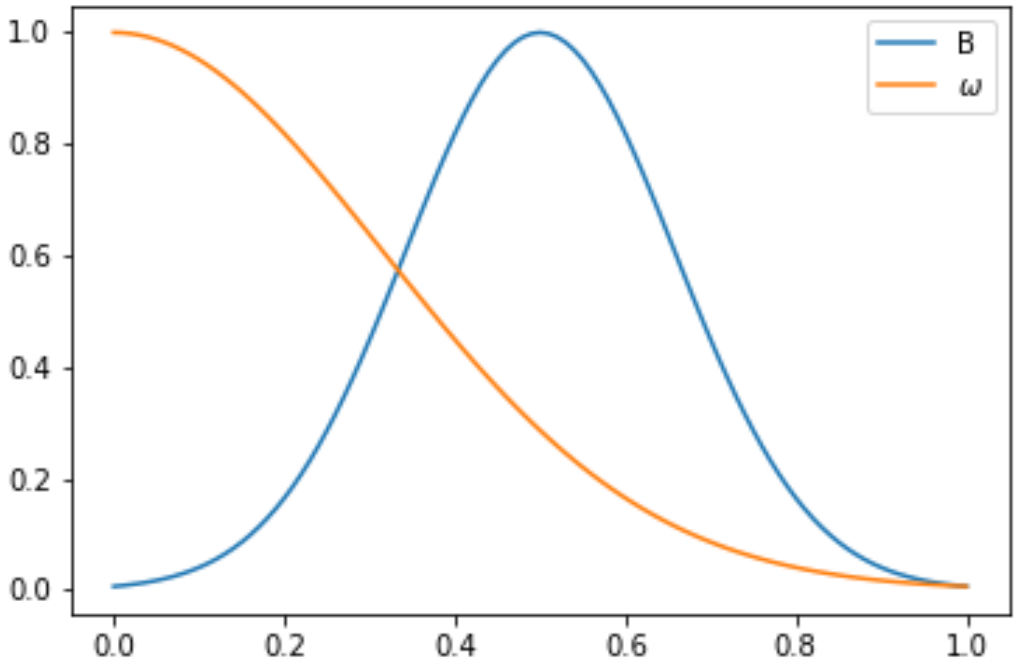


Figure 4.1: Mapping function for boundary distance (B) and mass center distance (ω) for parameter set: $\beta_B = 0.5$, $\beta_\omega = 0$, $\gamma_B = 20$, $\gamma_\omega = 5$.

The weighting function is a linear combination of the mapping function for the distances from the centroid and from the decision boundary:

$$w = \alpha f(dist_B, \beta_B, \gamma_B) + (1 - \alpha) f(dist_\omega, \beta_\omega, \gamma_\omega), \tag{4.2}$$

where $dist_B$ denotes the distance from the decision boundary and $dist_\omega$ – from the centroid, $0 \leq \alpha \leq 1$.

Formally, the proposed algorithm for classifier integration can be expressed as:

$$\Psi_\alpha(x, \beta_B, \beta_\omega, \gamma_B, \gamma_\omega) = \arg\max_{\omega_c} \sum_{k=1}^{K} I(\Psi_k(x), \omega_c)(\alpha f(dist_B(x), \beta_B, \gamma_B) + (1-\alpha) f(dist_\omega, \beta_\omega, \gamma_\omega)) \quad (4.3)$$

Algorithm 3: Classification algorithm based on distance from decision boundary and centroid.

Input: K – The number of base classifiers ($\Psi_1, \Psi_2, \ldots, \Psi_K$), α - share of the distance from decision boundary in the weight computation ($0 \leq \alpha \leq 1$), x - classified object

Output: ω_c - The label prediction

1 Divide the dataset into $K+1$ subsets (K for training base models and 1 for testing).
2 Using training subsets, calculate the centroids for each label using.
3 Train the base classifiers $\Psi_1, \Psi_2, \ldots, \Psi_K$.
4 Determine the label indicated by the decision tree for the classified object, then calculate the weight for it using the equation 4.2.
5 Sum the weights for the separate labels over all base classifiers.
6 Using the formula 4.3, assign to the object x the label with the largest sum of weights.

4.1.2. Results and analysis

In the experiments, the feature selection process was performed to reduce the number of features to the two most informative. Dimensionality reduction followed to avoid unnecessary complexity. Afterwards, decision trees as base classifiers were trained and a pool of classifiers consisting of five models was created.

The main goal of the experiments was to compare the quality of the classification of the proposed method of decision tree ensembling using the geometric representation with referrential techniques: majority voting (Ψ_{MV}) and random forest (Ψ_{RF}). The variable parameter α is written as a subscript in the classifier notation: Ψ_α. Four different values of α were studied: $\alpha \in \{0, 0.3, 0.7, 1\}$. Two classification measures: Matthews correlation coefficient (MCC) and accuracy (ACC) were used to conduct statistical comparison between the algorithms. Tables 4.2, 4.4, 4.6, 4.8 present the results of ACC and tables 4.3, 4.5, 4.7, 4.9 – of MCC. Along with the quality measures, the average ranks obtained in the nonparametric Friedman rank tests are presented in the last column of each table.

Table 4.2: ACC values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 20$ and $\gamma_\omega = 5$.

	bi	bu	c	d	h	io	me	po	se	wd	wi	rank
Ψ_{MV}	0.720	0.568	0.716	0.912	0.707	0.759	0.753	0.910	0.931	0.900	0.935	4.23
Ψ_{RF}	0.724	0.546	0.840	0.919	0.746	0.775	0.727	0.911	0.931	0.909	0.944	2.59
$\Psi_{0.0}$	0.715	0.658	0.742	0.915	0.717	0.736	0.777	0.915	0.938	0.906	0.955	2.73
$\Psi_{0.3}$	0.722	0.497	0.761	0.907	0.691	0.789	0.762	0.914	0.935	0.889	0.937	3.68
$\Psi_{0.7}$	0.726	0.528	0.769	0.907	0.725	0.730	0.763	0.902	0.926	0.907	0.953	3.68
$\Psi_{1.0}$	0.692	0.582	0.758	0.921	0.742	0.746	0.713	0.913	0.927	0.888	0.929	4.09

Table 4.3: MCC values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 20$ and $\gamma_\omega = 5$.

	bi	bu	c	d	h	io	me	po	se	wd	wi	rank
Ψ_{MV}	0.406	0.095	0.426	0.826	0.032	0.465	0.488	-0.002	0.022	0.791	0.853	3.82
Ψ_{RF}	0.415	0.078	0.691	0.836	0.169	0.512	0.468	0.000	-0.003	0.810	0.876	2.59
$\Psi_{0.0}$	0.435	0.304	0.514	0.828	0.171	0.417	0.484	0.000	0.000	0.801	0.896	2.73
$\Psi_{0.3}$	0.377	-0.026	0.605	0.815	0.072	0.460	0.501	0.000	-0.003	0.773	0.859	4.09
$\Psi_{0.7}$	0.405	0.059	0.589	0.817	0.023	0.491	0.524	0.000	0.000	0.804	0.890	3.18
$\Psi_{1.0}$	0.236	-0.067	0.505	0.843	0.072	0.445	0.437	0.000	0.000	0.767	0.841	4.59

Table 4.4: ACC values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 5$ and $\gamma_\omega = 5$.

	bi	bu	c	d	h	io	me	po	se	wd	wi	rank
Ψ_{MV}	0.720	0.568	0.716	0.912	0.707	0.759	0.753	0.910	0.931	0.900	0.935	4.41
Ψ_{RF}	0.724	0.546	0.840	0.919	0.746	0.775	0.727	0.911	0.931	0.909	0.944	3.05
$\Psi_{0.0}$	0.729	0.624	0.806	0.924	0.637	0.792	0.701	0.917	0.930	0.918	0.961	2.55
$\Psi_{0.3}$	0.685	0.593	0.800	0.910	0.723	0.769	0.624	0.899	0.933	0.901	0.942	4.41
$\Psi_{0.7}$	0.718	0.608	0.680	0.915	0.756	0.779	0.750	0.931	0.933	0.906	0.954	2.86
$\Psi_{1.0}$	0.737	0.494	0.738	0.917	0.768	0.710	0.683	0.926	0.935	0.864	0.928	3.73

Table 4.5: MCC values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 5$ and $\gamma_\omega = 5$.

	bi	bu	c	d	h	io	me	po	se	wd	wi	rank
Ψ_{MV}	0.406	0.095	0.426	0.826	0.032	0.465	0.488	-0.002	0.022	0.791	0.853	4.09
Ψ_{RF}	0.415	0.078	0.691	0.836	0.169	0.512	0.468	0.000	-0.003	0.810	0.876	2.50
$\Psi_{0.0}$	0.412	0.175	0.614	0.846	0.027	0.500	0.387	0.000	-0.007	0.827	0.917	2.64
$\Psi_{0.3}$	0.321	0.145	0.587	0.823	0.093	0.496	0.216	0.000	-0.005	0.790	0.873	4.27
$\Psi_{0.7}$	0.403	0.113	0.323	0.830	0.193	0.490	0.348	0.000	-0.003	0.793	0.899	3.59
$\Psi_{1.0}$	0.449	-0.010	0.477	0.838	0.126	0.428	0.378	0.000	0.000	0.710	0.831	3.91

Table 4.6: ACC values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 20$ and $\gamma_\omega = 20$.

	bi	bu	c	d	h	io	me	po	se	wd	wi	rank
Ψ_{MV}	0.720	0.568	0.716	0.912	0.707	0.759	0.753	0.910	0.931	0.900	0.935	3.77
Ψ_{RF}	0.724	0.546	0.840	0.919	0.746	0.775	0.727	0.911	0.931	0.909	0.944	2.68
$\Psi_{0.0}$	0.705	0.623	0.746	0.907	0.712	0.779	0.732	0.892	0.941	0.907	0.956	3.00
$\Psi_{0.3}$	0.711	0.571	0.881	0.907	0.694	0.794	0.671	0.878	0.929	0.899	0.959	3.77
$\Psi_{0.7}$	0.698	0.520	0.721	0.903	0.712	0.771	0.633	0.914	0.934	0.893	0.928	4.86
$\Psi_{1.0}$	0.702	0.573	0.740	0.911	0.735	0.787	0.747	0.903	0.937	0.914	0.931	2.91

Table 4.7: MCC values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 20$ and $\gamma_\omega = 20$.

	bi	bu	c	d	h	io	me	po	se	wd	wi	rank
Ψ_{MV}	0.406	0.095	0.426	0.826	0.032	0.465	0.488	-0.002	0.022	0.791	0.853	3.55
Ψ_{RF}	0.415	0.078	0.691	0.836	0.169	0.512	0.468	0.000	-0.003	0.810	0.876	2.55
$\Psi_{0.0}$	0.402	0.173	0.482	0.816	0.205	0.490	0.454	-0.027	0.033	0.801	0.897	3.00
$\Psi_{0.3}$	0.408	0.081	0.768	0.811	0.111	0.530	0.367	0.000	-0.003	0.790	0.910	3.09
$\Psi_{0.7}$	0.327	-0.039	0.448	0.804	0.083	0.460	0.282	0.000	0.000	0.777	0.841	5.09
$\Psi_{1.0}$	0.203	0.060	0.470	0.822	0.030	0.518	0.467	0.000	0.000	0.817	0.848	3.73

Table 4.8: ACC values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 10$ and $\gamma_\omega = 10$.

	bi	bu	c	d	h	io	me	po	se	wd	wi	rank
Ψ_{MV}	0.720	0.568	0.716	0.912	0.707	0.759	0.753	0.910	0.931	0.900	0.935	3.86
Ψ_{RF}	0.724	0.546	0.840	0.919	0.746	0.775	0.727	0.911	0.931	0.909	0.944	2.55
$\Psi_{0.0}$	0.746	0.632	0.745	0.915	0.730	0.752	0.779	0.907	0.931	0.904	0.925	3.14
$\Psi_{0.3}$	0.742	0.608	0.768	0.903	0.707	0.758	0.606	0.917	0.942	0.919	0.932	3.14
$\Psi_{0.7}$	0.734	0.528	0.848	0.914	0.693	0.781	0.643	0.889	0.928	0.895	0.925	4.32
$\Psi_{1.0}$	0.710	0.530	0.594	0.911	0.715	0.769	0.718	0.914	0.936	0.915	0.924	4.00

Table 4.9: MCC values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 10$ and $\gamma_\omega = 10$.

	bi	bu	c	d	h	io	me	po	se	wd	wi	rank
Ψ_{MV}	0.406	0.095	0.426	0.826	0.032	0.465	0.488	-0.002	0.022	0.791	0.853	3.86
Ψ_{RF}	0.415	0.078	0.691	0.836	0.169	0.512	0.468	0.000	-0.003	0.810	0.876	2.73
$\Psi_{0.0}$	0.434	0.310	0.508	0.829	0.231	0.497	0.592	-0.024	-0.004	0.797	0.825	3.59
$\Psi_{0.3}$	0.464	0.193	0.574	0.804	0.032	0.503	0.209	0.000	-0.003	0.827	0.845	3.41
$\Psi_{0.7}$	0.435	0.054	0.705	0.829	-0.017	0.544	0.294	0.000	0.000	0.785	0.839	4.59
$\Psi_{1.0}$	0.263	0.059	0.358	0.821	0.074	0.515	0.416	0.000	0.000	0.816	0.841	3.82

Bonferroni–Dunn tests were performed to check whether the examined algorithms behave differently. For the significance level of 0.1 the critical value of the difference between Friedman ranks for the test equals 1.44 (5 algorithms are compared with the reference method, 11 data sets are used). The figures 4.2 – 4.9 present the results of the statistical analysis. In the experiments, the integration algorithm outperforms the random forest only in the case of ACC for $\alpha \in 0, 0.7$ and $\gamma_B = \gamma_\omega = 5$ although the changes are not significant. Integrated classifier created with the parameter $\alpha = 0$ yields significantly better results than the majority voting for the parameter pairs: $\gamma_B = \gamma_\omega = 5$ (both ACC and MCC) and $\gamma_B = 20, \gamma_\omega = 5$ (ACC). Additionally, when $\gamma_B = \gamma_\omega = 5$, ACC of the integrated classifier is significantly better for $\alpha = 0.7$ than ACC of majority voting.

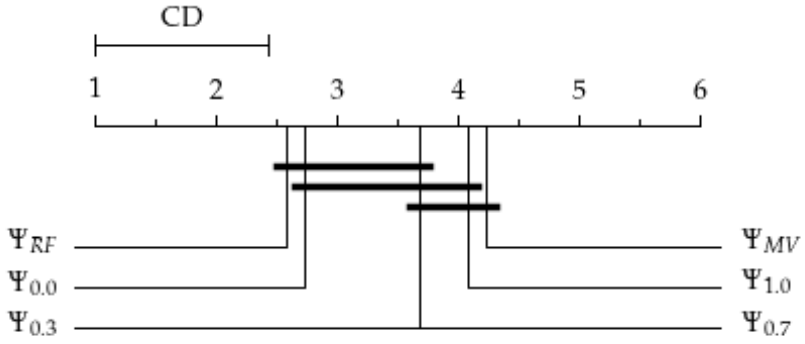


Figure 4.2: Friedman ranks comparison of the quality of the proposed algorithm, majority voting and random forest for ACC measure, $\gamma_B = 20$ and $\gamma_\omega = 5$.

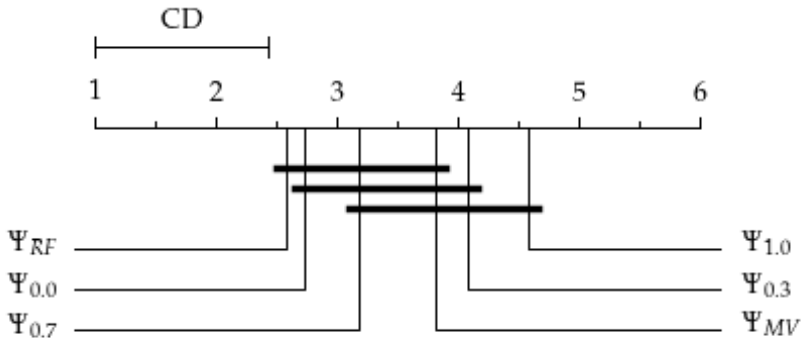


Figure 4.3: Friedman ranks comparison of the quality of the proposed algorithm, majority voting and random forest for MCC measure, $\gamma_B = 20$ and $\gamma_\omega = 5$.

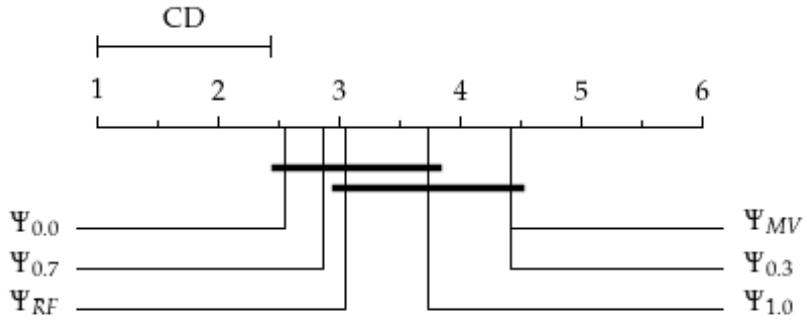


Figure 4.4: Friedman ranks comparison of the quality of the proposed algorithm, majority voting and random forest for ACC measure, $\gamma_B = 5$ and $\gamma_\omega = 5$.

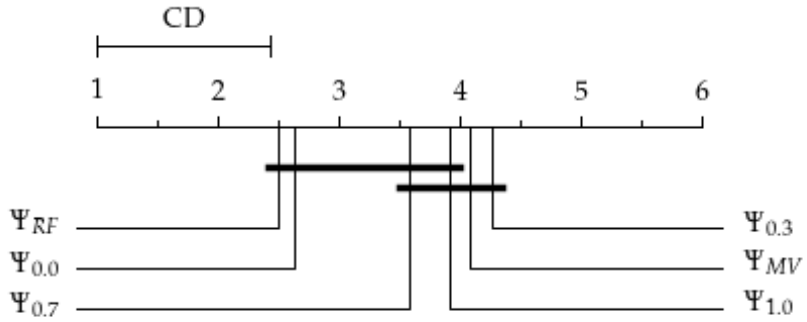


Figure 4.5: Friedman ranks comparison of the quality of the proposed algorithm, majority voting and random forest for MCC measure, $\gamma_B = 5$ and $\gamma_\omega = 5$.

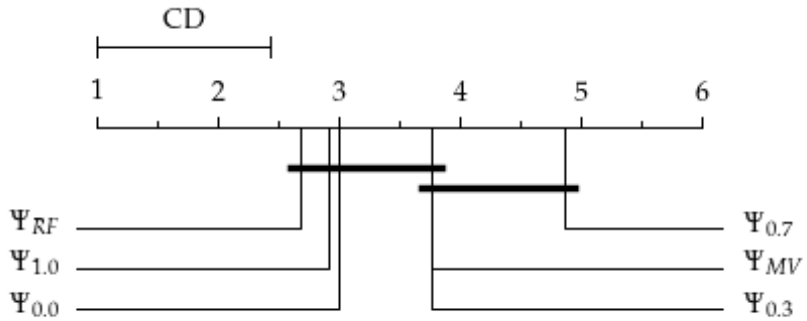


Figure 4.6: Friedman ranks comparison of the quality of the proposed algorithm, majority voting and random forest for ACC measure, $\gamma_B = 20$ and $\gamma_\omega = 20$.

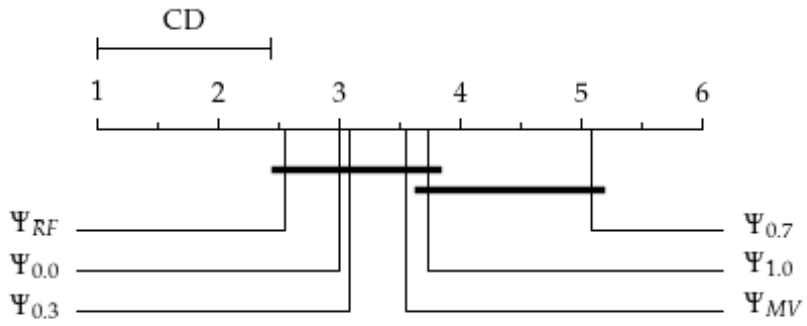


Figure 4.7: Friedman ranks comparison of the quality of the proposed algorithm, majority voting and random forest for MCC measure, $\gamma_B = 20$ and $\gamma_\omega = 20$.

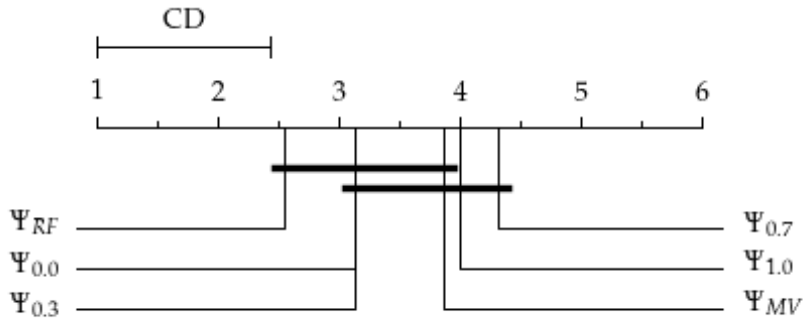


Figure 4.8: Friedman ranks comparison of the quality of the proposed algorithm, majority voting and random forest for ACC measure, $\gamma_B = 10$ and $\gamma_\omega = 10$.

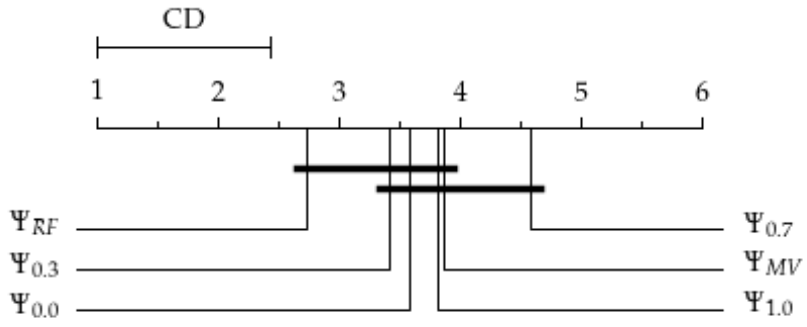


Figure 4.9: Friedman ranks comparison of the quality of the proposed algorithm, majority voting and random forest for MCC measure, $\gamma_B = 10$ and $\gamma_\omega = 10$.

4.1.3. Extended experiments

The published content presented is only a subset of the obtained results. Additionally, only a part of the datasets from the chapter 3 was used. The experiments were repeated for a greater number of datasets. This time the confusion matrices were saved to allow for computing more quality measures (see chapter 3). The obtained results can be found in the appendix A.

4.1.4. Conclusions

In the presented article, a novel approach to decision tree ensembling in the geometric space was proposed. The algorithm's definition is less restrictive than the majority voting regarding the base classifier count – no matter whether it is odd or even, the algorithm works without the need to resolve the ties. Additionally, the impact of the distance from the decision boundary and from the centroid defined by the center of mass was examined. Statistical tests were performed based on two quality measures: ACC and MCC.

The statistical analysis of the results containing two classification quality measures was conducted using 11 open–source datasets. Bonferroni–Dunn tests showed that, in most cases, the proposed algorithm for $\alpha = 0$ performs better than MV. The source code used for the experiments can be found on github: https://github.com/TAndronicus/dtree-merge-scoring.

4.2. Weighted Scoring in Geometric Space for Decision Tree Ensemble

In the paper [16] an algorithm was proposed that uses:

- Feature space division into disjoint subspaces.
- Horizontal partitioning of the dataset for training base models, in the proposal decision trees are used.
- The size and the location of the decision regions defined by the base classifiers as well as the location of the subspace's centroid are used to compute the weights further utilized in the integration phase.

The integration process in the proposed algorithm is based on the geometric representation. The label is determined for every region based on the mutual location between the region and the classification subspace.

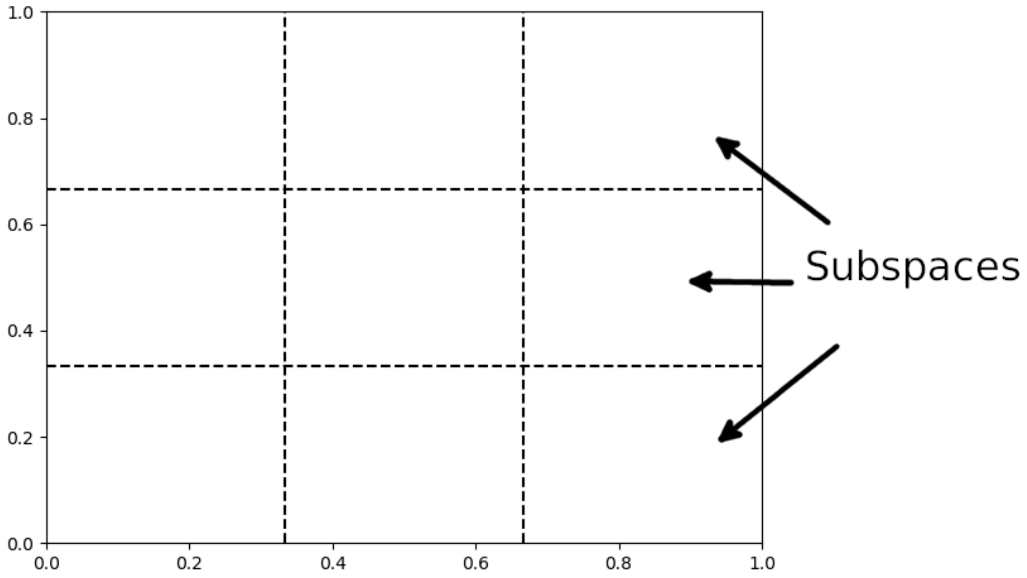
The objectives of the work are:

- A proposal of a new MCS algorithm that uses mutual location of the decision regions and the subspaces' centroids in the integration phase.
- Proving the hypothesis that the proposed algorithm is equivalent to majority voting for infinite granularity of space division.
- Experiments and statistical analysis for the evaluation of the proposed method.

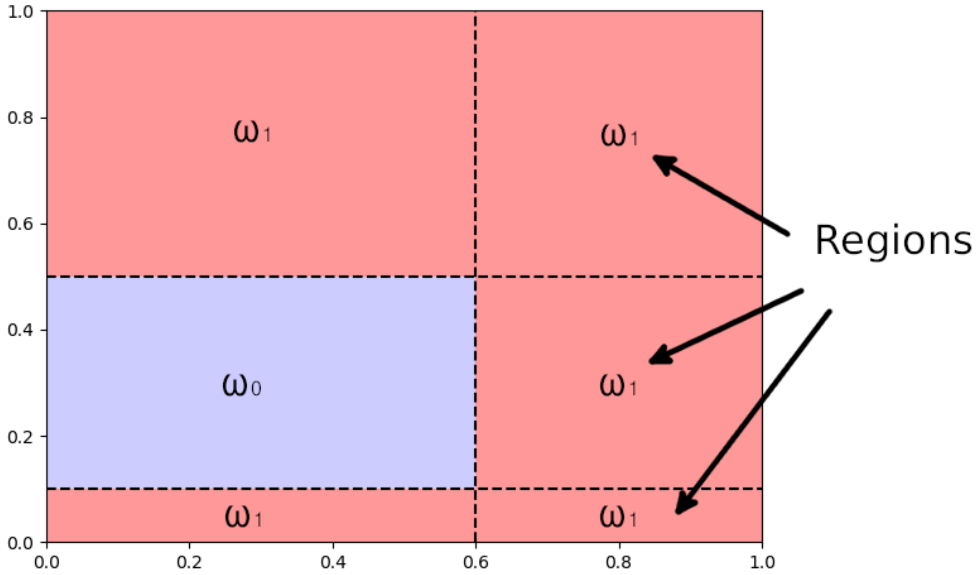
4.2.1. Proposed method

Decision tree divides the feature space into a set of cubes, each associated with a single class. In the article, a two-dimensional classification problem is considered. The decision boundaries can be represented as rectangles. The proposed algorithm ensembles multiple decision trees using this representation. Further on, these atomic areas will be referred to as classification regions.

The dataset (including training and testing subsets) generates a N–dimensional space. The static division is introduced and the space is split into disjunctive subsets. Those cubical sets (further referred to as subspaces) are of the same shape as the original dataset, since the cube is divided into the same number of parts along every dimension (feature axis) – they are scaled. *Subspaces* and *classification regions* are depicted in the figure 4.10. Three granularity degrees were studied: the edges were divided into 20, 40 and 60 segments. A middle point is determined for every region.



(a) *Subspaces*.



(b) *Classification regions*.

Figure 4.10: Graphical explanation of *subspace* and *classification region*.

The label of the classification region spanning over the midpoint of the specific subspace is selected as the label of the entire subspace. The weight for every candidate is calculated based on the function mapping the area (volume) of the classification region to a value in the range of $[0, 1]$. In the last step, the intermediate results are aggregated classification region–wise by summing their weights across the base models. Then the class with the largest aggregated weight is assigned to the subspace in the final model. Since the resulting classifier assigns labels to every rectangular region of competence, it is also a decision tree. The resulting classifier is also a decision tree, because it assigns a single label to every rectangular region of competence.

From the other perspective, the classification spaces are equal and the labels are determined by their middle points – the classification regions are therefore Voronoi cells. Hence, the resulting classifier can be considered as 1-NN. In this representation, the training objects of the NN classifier are located in the center points of every subspace. This simplifies the reasoning about the resulting model and eases its serialization. Note that the presented technique can be used as a cardinality reduction technique as an intermediate step of any machine learning algorithm. The centers of the subspaces can be passed as an input to another model. The resolution (or resulting cardinality) can be customized as any number of the form res^n, where $res \in \mathbb{N}_+$ and n denotes the desired dimensionality.

For the notation consistency, the model mapping the classification region (all objects that fall in the region) to the label will be denoted as $\Upsilon(A) \equiv \forall_{x \in A} \Psi(x)$. This assumes that all the objects falling in the classification region A must be classified with the same label by the classifier Ψ, like the Voronoi cells in decision trees or nearest neighbor classifiers.

Let R_l^k be the l-th classification region of k-th classifier having some label $\omega_i = \Upsilon(R_l^k)$ ($\omega_i \in \Omega$), S_m – m-th subspace and x – the object under test. Let us also denote by M the number of partitions of the classification space into subspaces along one dimension. This means that M^N subspaces will be generated for the N-dimensional problem. It is important to notice that the k-th decision tree can be completely represented using R^k. If we define $\delta_R(R_l^k, S_m)$ as 1 if the midpoint of S_m lies within R_l^k and 0 otherwise and $\delta_S(S_m, x)$ as 1 if S_m spans x and 0 otherwise, i.e.

$$\delta_S(S_m, x) = \begin{cases} 1 & \text{if } x \in S_m \\ 0 & \text{if } x \notin S_m \end{cases}$$

$$\delta_R(R_l^k, S_m) = \begin{cases} 1 & \text{if } mid(S_j) \in R_l^k \\ 0 & \text{if } mid(S_j) \notin R_l^k \end{cases}$$

and $f_m(R_l^k)$ as a weighting function, then the proposed algorithm can be written formally as:

$$\Psi_T(x) = \arg\max_{\omega_i} \sum_{k=1}^{K} \sum_{m=1}^{M^n} \sum_{l=1}^{|R^k|} \delta_S(S_m, x) \delta_R(R_l^k, S_m) f_{\omega_i}(R_l^k). \tag{4.4}$$

Let $mid(A)$ be the middle point of the cubic region A, the equation 4.4 can be rewritten as:

$$\Psi_T(x) = \arg\max_{\omega_i} \sum_{k=1}^{K} \sum_{m=1}^{M^n} \sum_{l=1}^{|R^k|} \delta(x, S_m, R_l^k) f_{\omega_i}(R_l^k), \tag{4.5}$$

where $\delta(x, S_m, R_l^k) = \delta_S(S_m, x) \delta_S(R_l^k, mid(S_m))$.

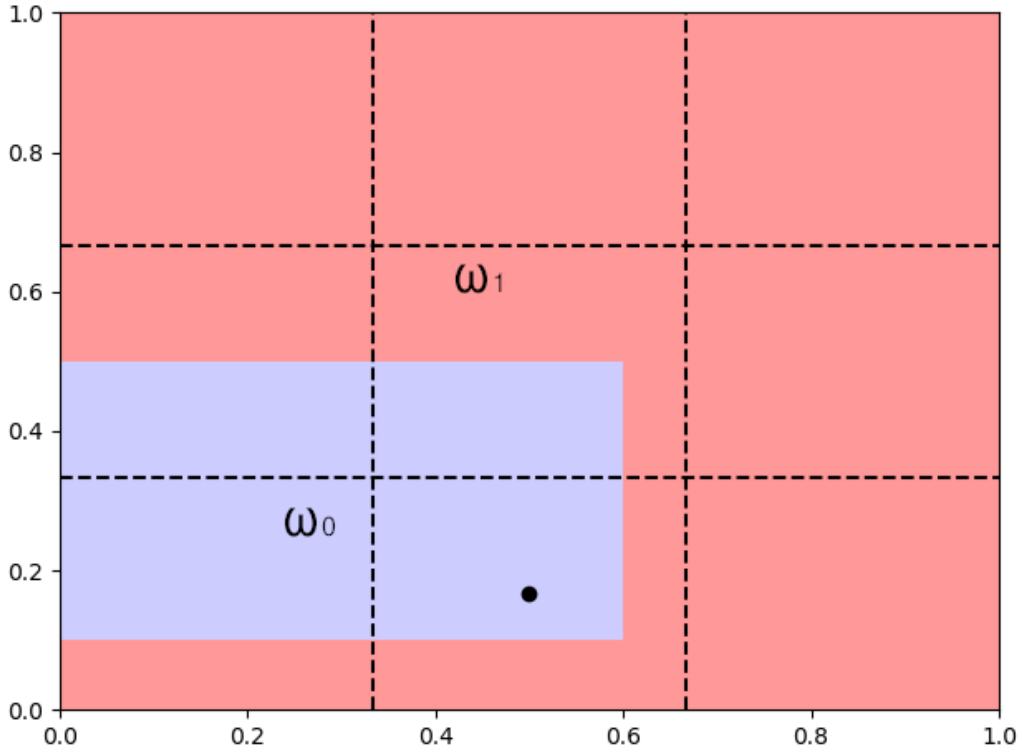


Figure 4.11: Label determination. The marked midpoint lies within a classification region labeled with ω_0, thus every object in this subspace is assigned label ω_0 with weight defined by the function 4.6.

The effective computational complexity is much lower as the majority of the terms of the equation 4.5 are omitted – in most cases either $\delta_S(S_m, x) = 0$ or $\delta_S(R_l^k, mid(S_m)) = 0$ holds.

The proportional and inversely proportional weighting functions were studied experimentally: $f_{vol}(A) = volume(A)$, $f_{inv}(A) = \frac{1}{volume(A)}$.

Another important observation is that f_{ω_i} depends on the label ω_i and because it assigns the weight to the label, it can be written more compactly as:

$$f_{\omega_i}(R_l^k) = f_{wt}(R_l^k) I(\Upsilon_k(R_l^k), \omega_i), \tag{4.6}$$

where $I(\cdot)$ is the indicator function from the equation 1.6 and $wt \in \{vol, inv\}$.

The examplary process of weight calculation is shown in the figure 4.11. Since the middle point of every subspace lies within the classification region classified with some label ω_0, hence the label assigned to the region is ω_0. This procedure resolves the labels and the weights are calculated using the formula 4.6.

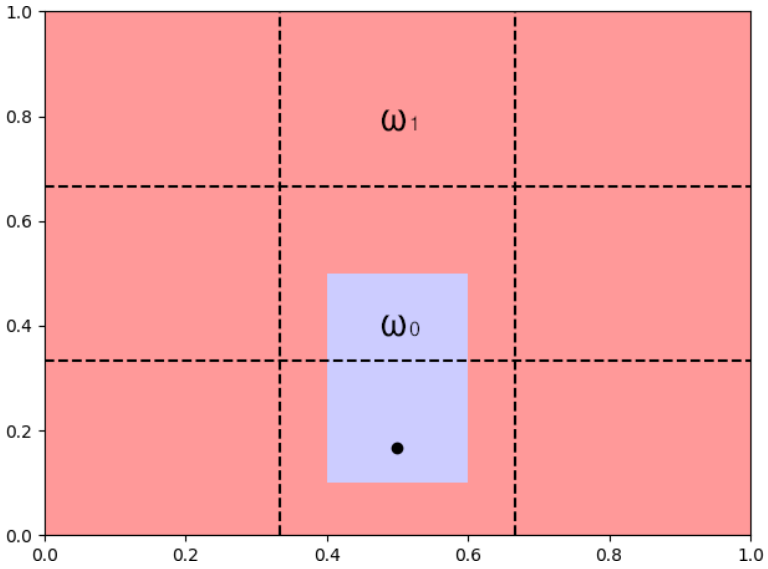
Algorithm 4: Algorithm to obtain the integrated decision tree using cubic subspaces.

Input: K – The number of the base models ($\Psi_1, \Psi_2, \ldots, \Psi_K$), M - the amount of splits along every feature axis, N - dimensionality of the problem

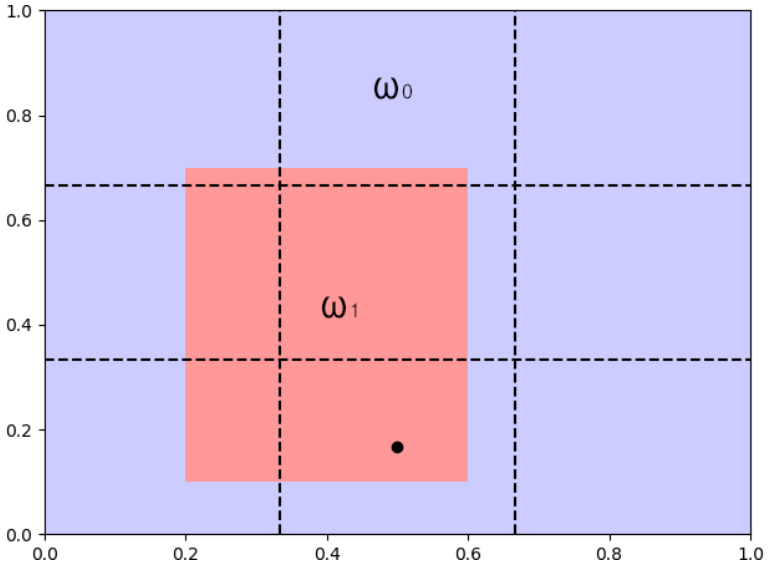
Output: Integrated decision tree Ψ_T

1 Normalize the dataset.
2 Choose the features that are the most informative.
3 Partition the dataset into $K + 1$ disjoint subsets (K for training base classifiers and 1 for testing).
4 Train the base classifiers $\Psi_1, \Psi_2, \ldots, \Psi_K$ and derive their geometrical representation (classification regions' coordinates with the corresponding labels).
5 Compute the coordinates of the M^n static classification regions.
6 For every subspace, determine the classification region that spans its midpoint and determine the competence of the label with the weighting formula 4.6.
7 Calculate the sum of the weights for every label over every classifier.
8 Assign classes to the subspaces by the greatest sum according to 4.5.

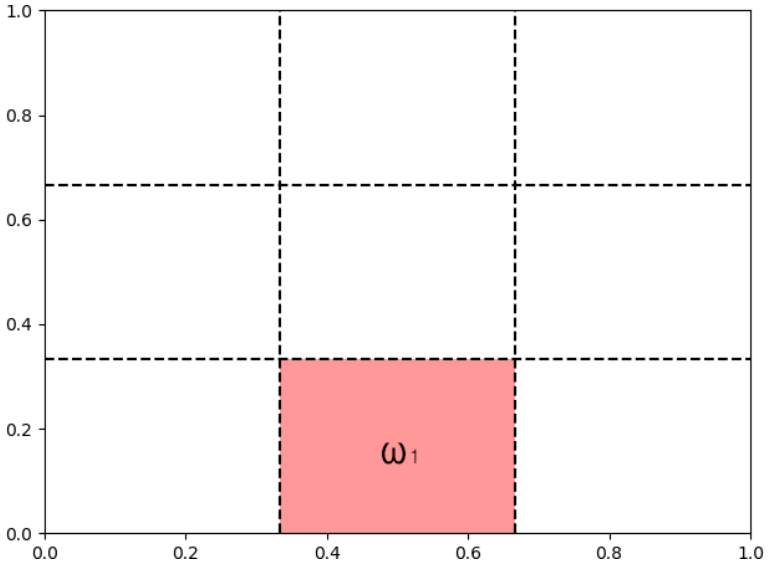
The algorithm is depicted in the figure 4.12. Given the subspace and the weighting function proportional to the classification subspace area, the first model assigns a weight to the label ω_0 and the second – to ω_1, because the middle points lie in the respective classification regions. The first label – ω_0 is assigned a smaller weight than the one associated with the second class – ω_1, because of the proportional mapping function used – the larger the area of the classification region, the larger the weight. The integrated model aggregates the weights and assigns the label ω_1 to the considered subspace as the label with the greatest sum of weights.



(a) First base classifier.



(b) Second base classifier.



(c) Resulting classification of subspace.

Figure 4.12: Label calculation using two base classifiers and mapping function proportional to area.

4.2.2. Generalization of majority voting

The problem of generalization of majority voting introduced in [16] was further developed in [13].

Lemma 1. *The (weighted) Majority Voting is a special case of the presented algorithm for infinitely dense space division.*

Proof. Firstly, let us notice that for any training point x, there is only one S_x and R_s^k that span over x (both classification regions and subspaces are disjoint).

The special case of the proposed algorithm is when the partitioning becomes infinitely dense. This means that the size of every subspace becomes infinitely small and shrinks to a single point:

$$\lim_{|S_x|\to 0} mid(S_x) = x \tag{4.7}$$

By combining the equations 4.7 and 4.4 the following formula can be obtained:

$$\lim_{M\to\infty} \Psi_T(x) = argmax_\omega \sum_{k=1}^{K} f_\omega(R_x^k), \tag{4.8}$$

where R_x denotes the decision tree region, that spans over x. This proves the lemma. □

In the special case of the function $f_\omega(R_x^k) = I(\Psi_k(x), \omega)$, where I is defined as in the equation 1.6, the (weighted) majority voting is obtained.

4.2.3. Results and analysis

A pool of models consisting of 5 decision trees having the maximal depth of 3 was created. As the reference classifiers, random forest and majority voting were used. The experiments were performed for edge split granularity of 20, 40 and 60 segments. M^N regions of competence were created when dividing the edge into M parts, where N is the number of dimensions. The algorithm is easily applicable to any number of dimensions without any modifications despite being examined using two–dimensional examples in the presented paper.

Two opposite weighting functions were studied: proportional and inversely proportional to the volume. The weight of every competence region of the decision tree can be calculated as its area (inverse of the area, respectively). Even influence of every dimension was assured by normalizing the datasets before the experiments.

Ψ_M^{weight} will denote the ensemble classifier, where $weight$ is the weighting function (vol stands for proportional to the regions' volume and inv – for inversely proportional to

the volume) and M - the number of divisions along every dimension. Reference methods are denoted as Ψ_{alg}, where alg stands for the algorithm used: random forest – RF or majority voting – MV. Tables 4.10 through 4.13 present the experimental results together with Friedman ranks.

Table 4.10: ACC and mean Friedman rank of majority voting, random forest and integrated classifiers for the proportional weighting function.

	bi	bu	c	d	h	io	me	po	se	wd	wi	rank
Ψ_{MV}	0.723	0.600	0.778	0.907	0.709	0.783	0.654	0.892	0.932	0.924	0.953	**2.32**
Ψ_{RF}	0.719	0.576	0.738	0.899	0.697	0.770	0.577	0.893	0.931	0.924	0.936	4.41
Ψ_{20}^{vol}	0.723	0.594	0.784	0.909	0.706	0.774	0.684	0.890	0.932	0.925	0.953	4.41
Ψ_{40}^{vol}	0.723	0.592	0.764	0.904	0.699	0.778	0.678	0.890	0.932	0.923	0.953	3.23
Ψ_{60}^{vol}	0.723	0.614	0.764	0.913	0.709	0.778	0.677	0.890	0.932	0.918	0.953	2.64

Table 4.11: ACC and mean Friedman rank of majority voting, random forest and integrated classifiers for the inversely proportional weighting function.

	bi	bu	c	d	h	io	me	po	se	wd	wi	rank
Ψ_{MV}	0.723	0.600	0.778	0.907	0.709	0.783	0.654	0.892	0.932	0.924	0.953	**2.23**
Ψ_{RF}	0.719	0.576	0.738	0.899	0.697	0.770	0.577	0.893	0.931	0.924	0.936	4.41
Ψ_{20}^{inv}	0.723	0.583	0.776	0.909	0.704	0.774	0.684	0.890	0.932	0.925	0.953	2.59
Ψ_{40}^{inv}	0.723	0.588	0.764	0.904	0.699	0.778	0.678	0.890	0.932	0.923	0.953	3.14
Ψ_{60}^{inv}	0.723	0.614	0.764	0.913	0.709	0.778	0.677	0.890	0.932	0.918	0.953	2.64

Table 4.12: MCC and mean Friedman rank of majority voting, random forest and integrated classifiers for the proportional weighting function.

	bi	bu	c	d	h	io	me	po	se	wd	wi	rank
Ψ_{MV}	0.415	0.169	0.583	0.815	0.140	0.567	0.328	-0.004	0.000	0.833	0.898	2.55
Ψ_{RF}	0.401	0.115	0.463	0.797	0.078	0.538	0.140	0.000	-0.001	0.831	0.860	4.50
Ψ_{20}^{vol}	0.415	0.166	0.594	0.819	0.141	0.543	0.378	-0.005	0.000	0.835	0.898	**2.32**
Ψ_{40}^{vol}	0.415	0.129	0.594	0.808	0.121	0.554	0.360	-0.005	0.000	0.831	0.898	3.14
Ψ_{60}^{vol}	0.415	0.205	0.594	0.826	0.140	0.554	0.364	-0.005	0.000	0.820	0.898	2.50

Table 4.13: MCC and mean Friedman rank of majority voting, random forest and integrated classifiers for the inversely proportional weighting function.

	bi	bu	c	d	h	io	me	po	se	wd	wi	rank
Ψ_{MV}	0.415	0.169	0.583	0.815	0.140	0.567	0.328	-0.004	0.000	0.833	0.898	**2.36**
Ψ_{RF}	0.401	0.115	0.463	0.797	0.078	0.538	0.140	0.000	-0.001	0.831	0.860	4.50
Ψ_{20}^{inv}	0.415	0.144	0.581	0.819	0.137	0.543	0.378	-0.005	0.000	0.835	0.898	2.68
Ψ_{40}^{inv}	0.415	0.142	0.594	0.808	0.121	0.554	0.360	-0.005	0.000	0.831	0.898	3.09
Ψ_{60}^{inv}	0.415	0.205	0.594	0.826	0.140	0.554	0.364	-0.005	0.000	0.820	0.898	**2.36**

To compare the improvement achieved when using the proportional and inversely proportional weighting function for decision trees with the referential methods, statistical tests were performed. Two classification quality measures were derived: accuracy and Matthews correlation coefficient.

As it was stated in 3.1, ACC is one of the most commonly used quantities, but it reflects the quality of the classifier trained on imbalanced datasets poorly. MCC takes the imbalance of the dataset into account, hence it is more reliable for the case of datasets used in the experiments, where the imbalance reaches 439.6.

The results of ACC are presented in the tables 4.10 and 4.11 and the results of MCC – in the tables 4.12 and 4.13. Along with the classification performance measures mentioned, Friedman average ranks are included as the last column.

Friedman tests' p-values are gathered in the table 4.14. Since none of the values exceed 0.01, not all algorithms perform equally. The odd algorithms can be determined using the post–hoc Bonferroni–Dunn tests.

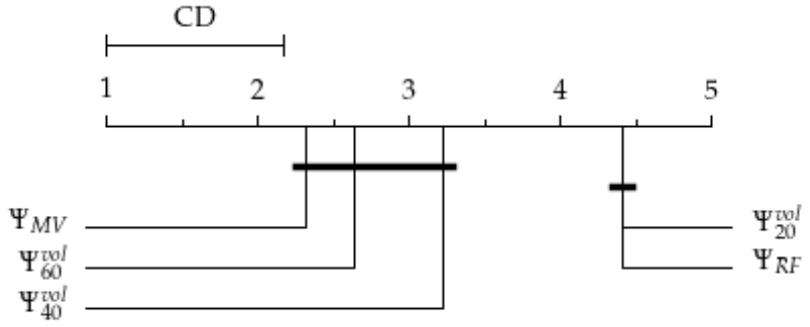
Table 4.14: p–values of ranked Friedman tests for the examined algorithms.

		Quality measure	
		ACC	MCC
Weighting	Proportional	0.004	0.006
	Inversely proportional	0.004	0.005

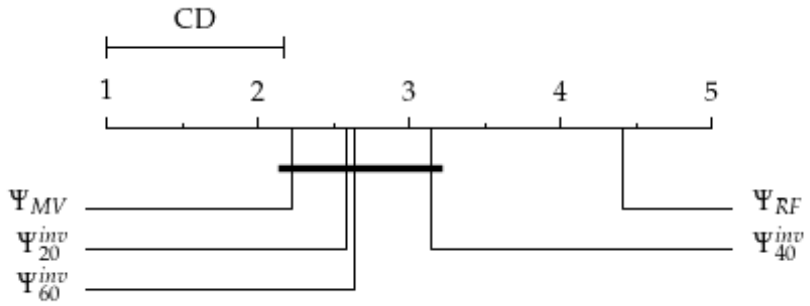
The tests were carried out for the entire set of feature space division (the feature space was divided into 20^2, 40^2 and 60^2 subspaces). The difference in the classification performance in several cases was observed. Bonferroni–Dunn tests require the difference in the Friedman ranks of at least 1.18 (11 datasets are used, 3 algorithms are compared with referential) to reject the null hypothesis with a significance level of $\alpha = 0.1$ [35].

Significantly different results are bolded. For every granularity of the feature space division and every measure, the ensembles composed using both weighting functions

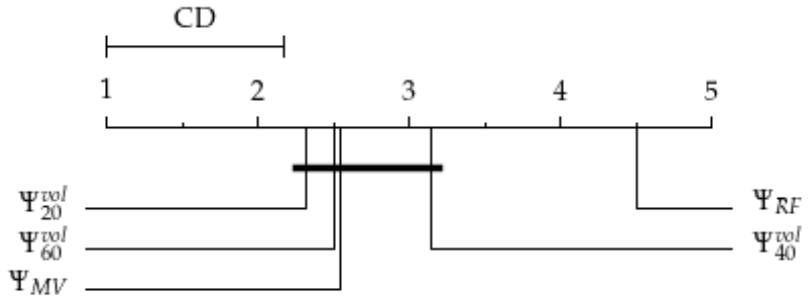
perform significantly better when compared to random forest. In addition to that, the proportional weighting function yielded better results than majority voting with respect to MCC.



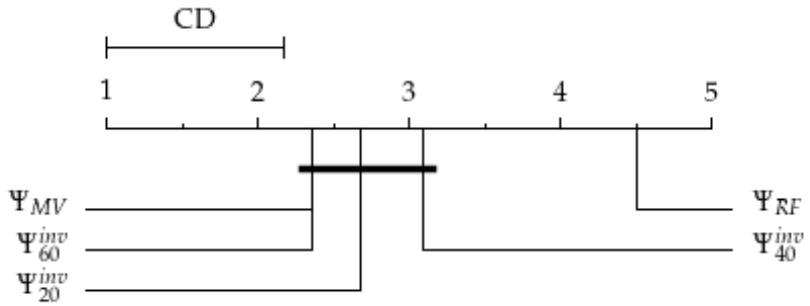
(a) Ranks of Friedman test of ACC for proportional weighting function.



(b) Ranks of Friedman test of ACC for inversely proportional weighting function.



(c) Ranks of Friedman test of MCC for proportional weighting function.



(d) Ranks of Friedman test of MCC for inversely proportional weighting function.

Figure 4.13: Friedman ranks comparison of the quality of the proposed algorithm, majority voting and random forest.

The statistical analysis of the results is depicted in the figure 4.13. The graphs present the ranks from the Friedman's tests. The higher the rank, the worse the performance of the model. The critical difference between the ranks was added in the top–left corner for the reference. The ranks within the critical segment are marked with the bolded line. They are indistinguishable according to Bonferroni–Dunn's test.

4.2.4. Extended experiments

Similarly as in the previous paper, the studies were rerun on a larger base of benchmarking datasets. The results are gathered in the appendix B. Additionally, the weighted MV was examined. For the weighted majority voting, Ψ_{wMV}^{weight} is used depending on the mapping function used. Intuitively, the results depend on the location of the subspaces, i.e. whether the corner of the subspace is in the point $(0, 0)$ or it is shifted (if the subspace lattice is shifted by $\frac{1}{M}$ along any feature axis, they are considered identical). The experiments concerning displacements of the lattice were conducted. If there are L displacements along the feature axis, then the following points are taken as the corner of the subspace: $(0,0), (0, \frac{1}{LM}), \ldots, (0, \frac{L-1}{LM}), (\frac{1}{LM}, 0), (\frac{1}{LM}, \frac{1}{LM}), \ldots, (\frac{L-1}{LM}, \frac{L-1}{LM})$ or in other words all the combinations of the form $\left(\frac{l_1}{LM}, \frac{l_2}{LM}\right)$, where $l_1, l_2 \in \{0, 1, \ldots, L-1\}$. The base studies (published in [16]) are the special case, where $l_1 = l_2 = 0$ or $L = 1$. For brevity this case will be referred to as with no displacements. They were presented in the section B.1. Section B.2 gathers the results for $L = 5$, where the results of all the combinations were averaged.

4.2.5. Conclusions

In the considered article, the algorithm of decision tree integration in the geometric space was proposed. Two opposing weighting functions were used to perform the experiments. The algorithm is less restrictive than majority regarding the number of base models. The possibility of a tie is negligible. The integrated model can be considered as a decision tree. This approach eases the reasoning about the model and its serialization.

All the subspaces are cubes, having the same shape as the feature space and can be considered Voronoi cells. Therefore, the resulting ensemble is a 1-NN classifier having centroids placed in the middle points of every subspace.

It was also proven that the weighted majority voting is a special case of the presented algorithm, when the division granularity is infinite.

11 open–source benchmarking datasets were used in the experimental part to perform the statistical analysis of the results regarding two classification measures. Together with ACC, MCC was calculated to take the highly imbalanced datasets into account. Bonferroni–Dunn tests indicated that for every granularity, the proposed algorithm resulted in a better classification quality than the random forest. In addition, MCC for the proportional weighting function of the ensemble outperformed the random forest. The source code used to conduct the experiments is hosted on github: https://github.com/TAndronicus/dtree-merge.

4.3. Decision Tree Integration Using Dynamic Regions of Competence

The further studies were in the area of dynamic regions of competence in decision tree ensembles. The first results in this topic were published in [14]. In this paper, the algorithm using partitioning of the feature space whose split point is determined by the decision rules of all decision tree nodes. The centroids of the new subspaces are found after division and influence the weights needed in the integration phase where the weighted majority voting rule is applied. The experiments involving several open–source benchmarking datasets demonstrate the effectiveness of the proposed method when compared with other MCS approaches. To discuss the results of the experiments, micro– and macro–average classification performance measures are used.

4.3.1. Proposed method

In this work, a novel approach of feature space division into disjoint subspaces is proposed. In this proposal, the partitioning process follows base classifier learning as opposed to the clustering and selection methods. No clustering to define a feature subspace is used in this work. The partitioning of the feature space is determined by the base classifier models exactly through their decision boundaries. Finally, the centroids of the resulting regions are used in the weighted majority voting to extract the final MCS decision.

To summarize, the main objectives of this work are as follows:

- A proposal of a new feature space partitioning method whose split is determined by the decision boundaries of each decision tree.
- A design of a new weighted majority voting rule algorithm dedicated to the fusion of decision tree models.
- Experimental studies to compare the proposed method with other MCS approaches employing diverse performance measures.

While the articles presented in the sections 4.1 and 4.2 used static division into regions of competence, this paper presents an algorithm with a dynamic approach. The main goal of introducing dynamically generated Voronoi cells is to improve the classification performance in comparison to using referential methods of decision tree committee ensembling: random forest and majority voting.

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

The first step of the presented algorithm is training a pool of base models – decision trees. To make sure that the base trees are indeed different from one another, training on the random subsets of the dataset is performed. Having a committee of trained decision trees, rectangular regions fulfilling the following properties are extracted:

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

Intuitively, 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 from every base model's perspective.

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

$$f(\Psi_i, s_0) = \frac{1}{\sigma} \sum_{s \in S} c_{s,\Psi_i}(1 - d(x_{s_0,mid}, x_{s,mid}))\delta(s_0, s) + \frac{c_{s_0,\Psi_i}}{2n} \tag{4.9}$$

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

$$\delta(s_0, s) = \begin{cases} 1 & \text{if } x_{s_0,1,min} = x_{s,2,max} \text{ or } x_{s_0,1,max} = x_{s,1,min} \\ & \text{or } x_{s_0,2,min} = x_{s,2,max} \text{ or } x_{s_0,2,max} = x_{s,2,min} \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

There is an important observation to be done that given the formula 4.10, the equation $\delta(s, s) = 0$ holds for every subspace s. This convention was chosen because the subspace's influence is contained within the second summand of the equation 4.9. The term $2n$ was chosen to normalize the term. As a result, the values for the subspace s_0 sum up to half of the weight's value, i.e.:

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

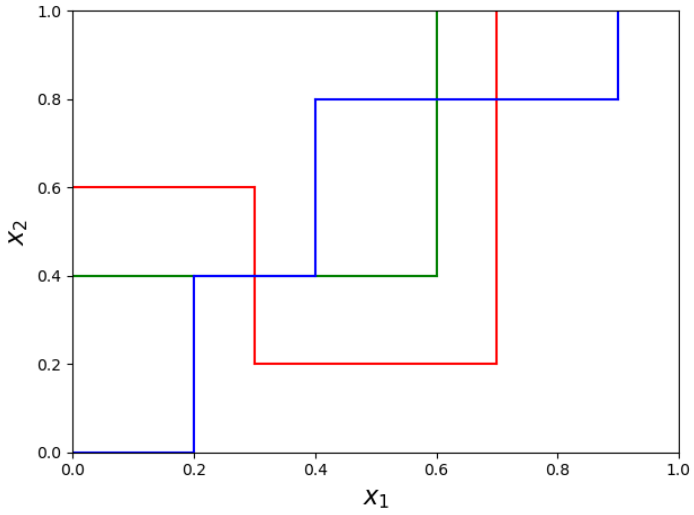
The figure 4.14 depicts the process of obtaining subspaces. Suppose that the base models (marked with colorful lines on the subfigure a) are equally oriented. Graphically, this means that all the points below the decision boundary are labelled by this model

with a single label, different from all the objects above the line (a binary classification problem is considered). The regions of competence are generated by splitting the feature space at the splitpoints of all the base decision trees (subfigure b). Computing the weight of the label for each region involves the region itself (filled with dark grey in subfigure c) together with its neighbors (lightgrey in subfigure c). Contributions from the neighbors depend on the distance between its middle point and the middle point of the region the weight is computed for. The algorithm 5 summarizes the procedure.

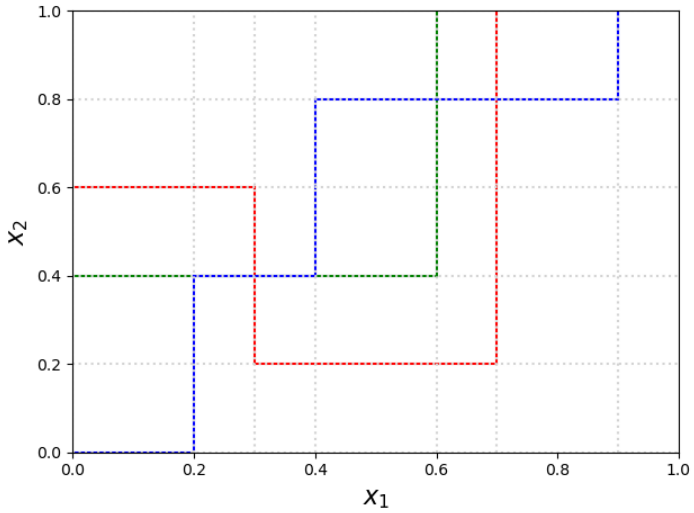
Algorithm 5: Classification algorithm using dynamic regions of competence obtained from decision trees.

Input: K – number of base classifiers ($\Psi_1, \Psi_2, \ldots, \Psi_K$)
Output: Integrated decision tree Ψ_i

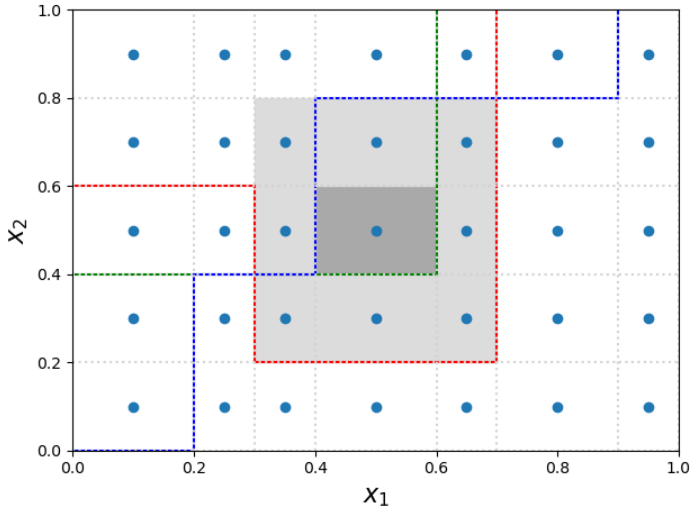
1 Normalize the dataset and select the two most informative features.
2 Split the dataset into $K + 1$ subsets (K for training every base decision tree and 1 for testing).
3 Train the base classifiers $\Psi_1, \Psi_2, \ldots, \Psi_K$ and obtain their geometrical representation (splits and labels).
4 Divide the feature space using splits of all the decision trees.
5 For every region and every label calculate the weight using the formula 4.9.
6 Classify every region by picking the label with the highest weight value.



(a) Base classifiers.



(b) Classification regions.



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

Figure 4.14: The process of extracting subspaces from base classifiers and determining neighbors for a subspace.

4.3.2. Results and analysis

$\mathrm{K} = 3$ was set as the number of base classifiers in the experiments. The studies were conducted in order to compare the classification quality measures observed when using the proposed algorithm (with the subscript I) and the commonly used referential methods: random forest (subscript RF) and majority voting (subscript MV). The experiments were performed on multiclass datasets, so micro– and macro–average precision, recall and F-score (the harmonic mean of precision and recall) were used. The F-score was computed, because of the high imbalance of multiple datasets used, as it was indicated in the section 3.1. Additionally, the overall accuracy was presented. Table 4.15 shows the results of average accuracy, micro– and macro–averaged F-score and tables 4.16, 4.17 – of precision and recall, micro– and macro–averaged respectively. Together with the metrics, Friedman ranks are presented – the larger the rank, the worse the classifier performs. Some of the micro–average performance measures of precision and recall are equal, because of the frequent single-label per instance problem ($\mathrm{Precision}_\mu = \mathrm{Recall}_\mu$) [129].

Considering all the calculated classification performance measures, the integrated classifier outperforms the referrential algorithms as indicated by the Friedman ranks. Post–hoc Nemenyi test after Friedman ranking requires the difference in ranks of at least 0.38 to falsify the null hypothesis of the algorithms being equivalent. This condition is met for $\mathrm{F} - \mathrm{score}_\mu$, which means that the proposed method – Ψ_I achieves statistically better results than the references: Ψ_{RF} and Ψ_{MV}. The micro–average measure is the fraction of the instances predicted correctly over all classes, hence the micro–average can be considered a more reliable metric than macro–average in the imbalanced datasets. Thus, the results indicate the improvement in the recognition, in particular for imbalanced data. For micro–average precision and recall the difference in ranks between Ψ_I and Ψ_{RF} showed statistically significant differences in the results. In the case of macro–average precision, the obtained results do not indicate a significant difference, whereas for macro–average recall (see table 4.17) the obtained average Friedman ranks are equal for Ψ_I and Ψ_{RF} algorithms.

Table 4.15: Average accuracy and F-scores for the random forest, the majority voting and the proposed algorithm together with Friedman ranks.

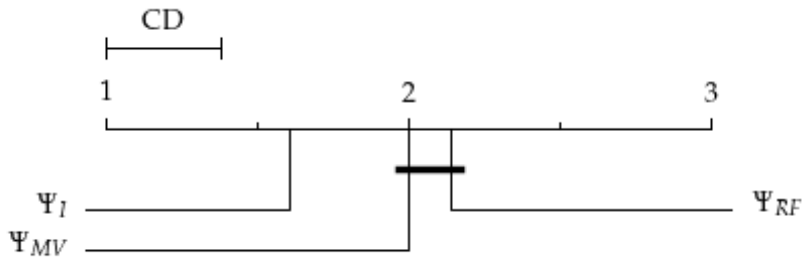
	Average accuracy			$\mathrm{F-score}_{\mu}$			$\mathrm{F-score}_{M}$		
Dataset	Ψ_{MV}	Ψ_{RF}	Ψ_{I}	Ψ_{MV}	Ψ_{RF}	Ψ_{I}	Ψ_{MV}	Ψ_{RF}	Ψ_{I}
aa	0.917	0.918	0.919	0.469	0.474	0.477	0.196	0.192	0.176
ap	0.853	0.812	0.863	0.853	0.812	0.863	0.676	0.559	0.692
ba	0.789	0.808	0.815	0.683	0.712	0.722	0.483	0.493	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 4.16: Micro–average precision and recall for the random forest, the majority voting and the proposed algorithm together with Friedman ranks.

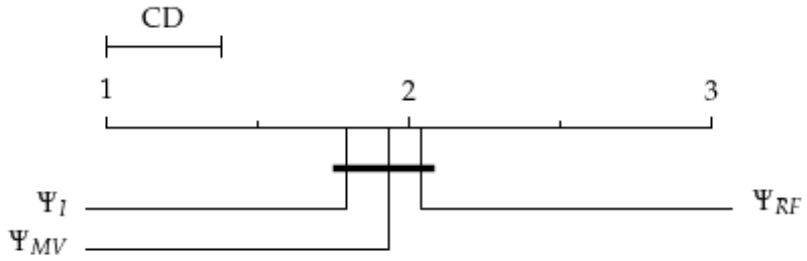
	Precision_μ			Recall_μ		
Dataset	Ψ_{MV}	Ψ_{RF}	Ψ_I	Ψ_{MV}	Ψ_{RF}	Ψ_I
aa	0.469	0.475	0.477	0.469	0.473	0.477
ap	0.853	0.812	0.863	0.853	0.812	0.863
ba	0.683	0.712	0.722	0.683	0.712	0.722
bi	0.736	0.736	0.702	0.736	0.736	0.702
bu	0.579	0.527	0.536	0.579	0.527	0.536
c	0.762	0.867	0.684	0.762	0.867	0.684
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 4.17: 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



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



(b) Friedman ranks with the critical Nemenyi test value for $\mathrm{F-score}_{M}$.

Figure 4.15: Friedman ranks comparison of the quality of the proposed algorithm, majority voting and random forest.

4.3.3. Conclusions

In these studies, a new approach for MCS creation was presented. Contrary to the clustering and selection method, the feature space partition in the proposal is based on the decision boundaries defined by the base classifier models. This involves the trained base classification models instead of clustering to determine the feature subspace. The centroids of the subspaces 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 regions of competence.

The experimental results indicate that the proposed method may generate an ensemble classifier that outperforms the commonly used methods of combining decision tree models — majority voting and random forest. The results show that the proposed method statistically improves the classification performance measured by the micro–average quantities. The source code written to perform the experiments is hosted on github: https://github.com/TAndronicus/dynamic-dtree.

4.4. Integration of Decision Tree Models Using the Decision Boundaries Defined by These Trees

Further research was done in the field of decision tree integration using dynamic regions of competence. As of this writing, the following work is in the middle of the publication process. The presented algorithm extends our previous approach that does not use classical clustering methods to partition the feature space. The decision boundaries defined by decision trees are used to divide the feature space. Selected subregions of the feature space are used to determine the weights needed in the integration phase based on the weighted majority voting rule. The proposal was compared with other state-of-the-art ensemble methods and with the previous proposal. To discuss the results of our experiments, seven classification performance measures were used.

The algorithm presented in this work is an improvement of the algorithm presented in the previous work. The novel elements introduced in this method are:

- Bagging is used to provide distinct data for training.
- The distance between the regions is calculated between the average points of the validation objects inside the region.
- The regions are filtered based on the imbalance ratio of the validation objects within the region.
- Depending on the outcome of the filtering in the previous point, larger rings of regions can be taken into account when calculating the weight of the label for each region.

4.4.1. Proposed method

The method relates to the previous work in that sense that it utilizes the division of the classification space into dynamic regions of competence in order to integrate decision trees. The goal of dynamically generated Voronoi cells was to improve the classification quality in comparison with other multiple classifier systems of decision trees: majority voting and random forest.

To reduce the impact of the magnitude of the datasets' features, the normalization to the unit cube was conducted. Every attribute takes values from the range of $[0, 1]$. The two most informative features were selected using ANOVA method.

Firstly, the committee of decision trees was trained. A bagging procedure was followed to ensure the classifiers are different. As in the previous studies, the regions of competence are obtained dynamically based on decision trees' geometrical representation. They fulfill the following rules:

- Their area is maximal.

- They span only single-labelled points from the perspective of every model.

Those requirements split the entire space along every axis at all the split points of the base classifiers. As a consequence, the regions are labeled with the same class by every tree.

Every region has a representative point (further referred to as a midpoint). The coordinates of the midpoints are calculated as the means of the respective coordinates of all the objects from the validation dataset that fall into this region. These points will define the distance between the regions in the algorithm.

The notation is analogous to the one from the previous article. The set of obtained subspaces will be denoted as S and the range of subspace s along axes x_1 and x_2 by $(x_{s,1,min}; x_{s,1,max})$ and $(x_{s,2,min}, x_{s,2,max})$. Let us also denote the midpoint of the subspace s as $x_{s,mid}$. Two regions: s_1 and s_2 are in a neighbor relation (denoted as $s_1 \sim s_2$) if one of the following holds: $x_{s_0,1,min} = x_{s,2,max}$, $x_{s_0,1,max} = x_{s,1,min}$, $x_{s_0,2,min} = x_{s,2,max}$ or $x_{s_0,2,max} = x_{s,2,min}$. In other words, for all adjacent regions s_1 and s_2 the relation $s_1 \sim s_2$ holds.

The weights of the labels in every region are calculated using the following formula:

$$f(\omega_i, s_0) = \frac{1}{\sigma} \sum_{s \in S} c_{s,\omega_i}(1 - d(x_{s_0,mid}, x_{s,mid}))\delta(s_0, s) + \frac{c_{s_0,\omega_i}}{2n} \tag{4.11}$$

where $d(p_1, p_2)$ denotes the Euclidean distance between the points p_1 and p_2, $\Omega \ni \omega_i$ denotes one of the possible labels, c_{s,ω_i} is the count of models that classify the subspace s with the label ω_i, σ is the correction used to make the sum of weights equal to 1 and $\delta(s_0, s)$ is a neighbor function.

The neighbor function is determined in the following way. First, an intermediate value δ' is calculated for every region:

$$\delta'(s_0, s) = \begin{cases} 1 & \text{if } s_0 \sim s \text{ and } \gamma(s) \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

where $\gamma(s)$ is true only if the minor class of validation set objects that fall into this region make up at least 5% of the objects. Visually, the ring of regions around s_0 is examined for the imbalance ratio. Formally:

$$\gamma(s) = \begin{cases} \text{true} & \text{if } \forall_{\omega_i \in \Omega} \min(\frac{|s \ni o: \text{label}(o) = \omega_i|}{|s|}) \geq 0.05 \\ \text{false} & \text{otherwise} \end{cases} \tag{4.13}$$

where $\omega_i \in \Omega$ denotes the set of classes of the dataset and $\text{label}(o)$ returns the label of the object o.

If $\forall_{s \in S} \delta'(s_0, s) = 0$, then the second intermediate parameter is calculated:

$$\delta''(s_0, s) = \begin{cases} 1 & \text{if } (\exists\, S \backslash \{s_0\} \ni s' :\ s_0 \sim s' \text{ and } s' \sim s) \text{ and } \gamma(s) \\ 0 & \text{otherwise} \end{cases} \tag{4.14}$$

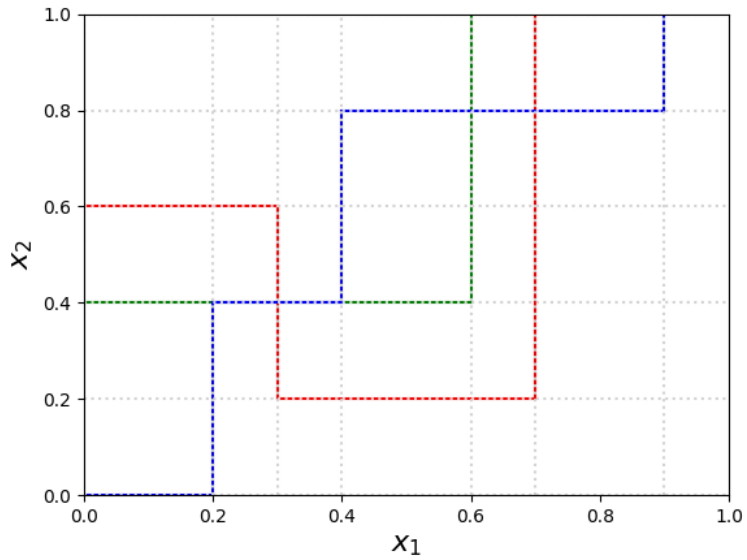
If $\forall_{s \in S} \delta''(s_0, s) = 0$, then the procedure is repeated analogously until some $\delta^n = 1$ is found or the entire classification space is examined. The formulas 4.12 and 4.14 indicate that $\delta'(s, s) = \delta''(s, s) = \cdots = \delta^n(s, s) = 0$ for every subspace s.

The contribution of the subspace itself is included in the second summand of the equation 4.11. Summing the weights coming from the regions results in 50% of the entire contribution, i.e.:

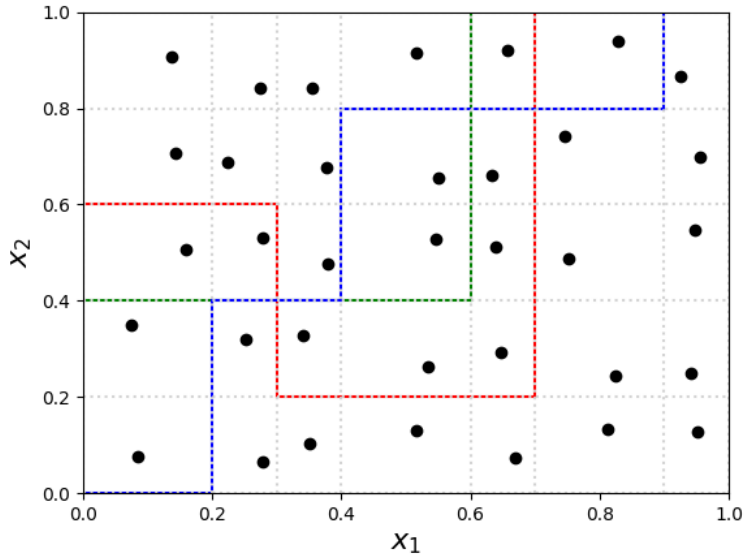
$$\sum_{i=1}^{n} \frac{c_{s_0, \Psi_i}}{2n} = \frac{1}{2}$$

The procedure of obtaining the regions is depicted in the figure 4.14. First, the dataset is prepared and the pool of decision trees is trained. As stated before, the competence regions are obtained by splitting the entire space at the split points of all decision trees (subfigure a). For each region, a midpoint is calculated (subfigure b). If any of the objects from the validation dataset fall into the region, their mean is taken as the midpoint, otherwise, a geometrical midpoint is calculated. This procedure is conducted for every region. When calculating the weight of the label for the region, the lookup throughout the neighbor regions is done to find the ones who have the minor class with the level of at least 5%. In the subfigure c none of the regions fulfill the requirement because of the high imbalance ratio (or no objects at all), so another outer ring of regions is taken into consideration (subfigure d). In the outer ring, the regions with a sufficient amount of objects of both classes were found (highlighted with darkgrey) and these objects are taken into consideration in the label weight calculation in equation 4.11. This procedure is conducted for every region.

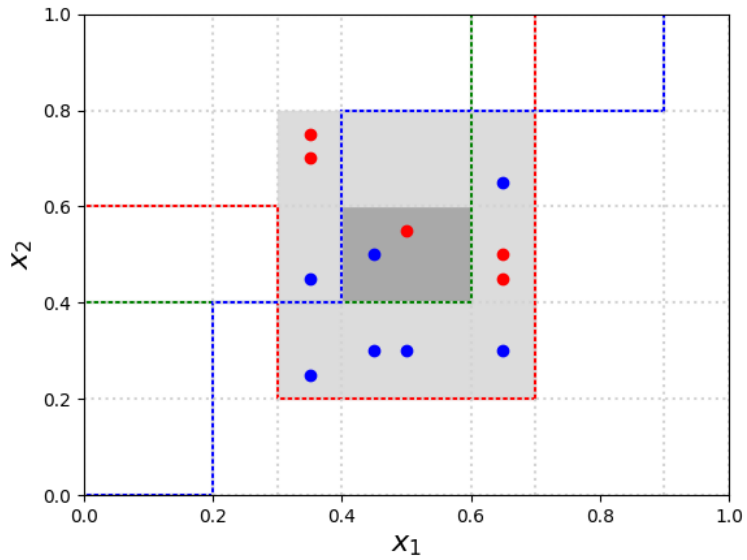
The entire procedure is presented in the algorithm 6.



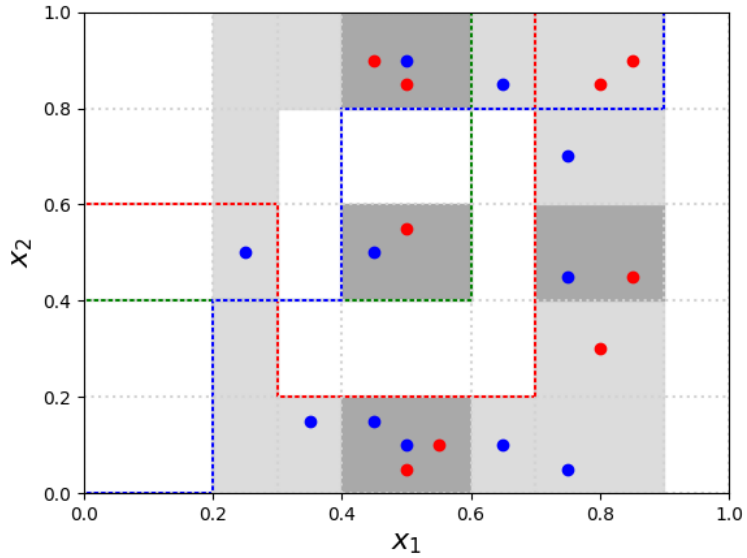
(a) Classification regions extracted using the splits of base classifiers.



(b) The entire decision space split into regions of competence with calculated midpoints.



(a) The first iteration of label weight calculation.



(b) The second iteration of label weight calculation.

Figure 4.16: The decision tree integration process. Colorful lines depict decision trees, blue and red dots - validation objects (od two different classes) and black dots - midpoints of the regions.

Algorithm 6: Algorithm for integration of decision trees using dynamic regions of competence.

Input: number of base classifiers ($\Psi_1, \Psi_2, \ldots, \Psi_K$) – K, dataset.
Output: Integrated decision tree – Ψ_I.

1 Normalize the considered dataset to the unit cube and select the two most informative numeric, non–categorical features.
2 Split the dataset into 3 subsets (for training - $\frac{K}{K+2}$ of the dataset's size, validation - $\frac{1}{K+2}$ of the dataset's size and testing - $\frac{1}{K+2}$ of the dataset's size).
3 Generate K collections of elements from the first subset using bagging for base classifiers training.
4 Train the base classifiers $\Psi_1, \Psi_2, \ldots, \Psi_K$ and extract the regions from their geometrical representation together with the associated labels for each model.
5 Calculate the midpoints of each region as the average of all the objects from the validation dataset that fall into this region.
6 Using the formula 4.11 determine the weight for every label in all regions of the classification space.
7 Assign the label with the highest weight to every region, i.e.: $\forall_{o \in s} \Psi(o) = \arg\max_{\omega} f(\omega, s)$.

4.4.2. Results and analysis

The integrated classifiers were evaluated against referential methods of ensemble classification algorithms: majority voting of the base classifiers and random forest. The average results were presented. Different numbers of base classifiers were examined: $K \in \{3, 5, 7, 9\}$.

The experiments were conducted in order to compare the classification quality of the previously presented algorithm (with IO subscript), the improved version presented here (with I subscript) and referential ensemble techniques: random forest and majority voting. The micro– and macro–average precision, recall, and F-score were utilized in order to evaluate the quality of the algorithms on multiclass problems. Overall accuracy is presented alongside the named measurements. The F-score was used because of the high imbalance ratio of some of the datasets (see section 3.1). It describes the quality of an algorithm better than the accuracy if tested on a nonbalanced dataset. Accuracy can be high if all the objects are labeled with the major label, although the model does not differentiate between the classes. The results of the experiments are presented in the table 4.18 (average accuracy), 4.19 (F-score), 4.20 and 4.21 (micro– and macro–average F-score respectively). The last row presents Friedman ranks – the smaller the rank, the better the quality of the model. All the results presented here were obtained for the nine base classifiers, because for that count they were the most promising.

The calculated Friedman ranks indicate that the proposed method outperforms other algorithms. This applies to all the calculated metrics, i.e., average accuracy, precision, recall and F-score. Bonferroni–Dunn post–hoc test confirms the hypothesis. According to [35] the difference between the rank of an integrated algorithm and any other must be greater than the critical value of 0.75 to discard the null hypothesis that the algorithms yield the same result at the confidence level $\alpha = 0.1$ (4 algorithms, 27 datasets).

The micro–average measure counts the fraction of instances correctly predicted across all classes. For this reason, the micro-average can be a more useful metric than the macro–average in a highly imbalanced dataset.

The partial results also suggest that the presented method improves the classification quality. This is especially visible for imbalanced datasets. This conclusion is confirmed by the difference in ranks between micro–average precision and recall. The critical difference graphs of Bonferroni–Dunn tests for nine base models for average accuracy, $\mathrm{F-score}_M$ and $\mathrm{F-score}_\mu$ are presented in the figure 4.17.

Table 4.18: Average accuracy of the majority voting, random forest, the proposed algorithm and the improved one along with Friedman ranks.

	Average accuracy			
Dataset	Ψ_{MV}	Ψ_{RF}	Ψ_{IO}	Ψ_{I}
ap	0.894	0.777	0.842	0.953
ba	0.795	0.790	0.795	0.803
bi	0.723	0.723	0.661	0.718
bu	0.544	0.546	0.558	0.820
c	0.839	0.879	0.839	0.890
d	0.924	0.942	0.925	0.969
e	0.762	0.766	0.762	0.857
h	0.729	0.719	0.760	0.846
io	0.808	0.830	0.762	0.866
ir	0.973	0.990	0.973	0.987
ma	1.000	1.000	1.000	1.000
me	0.534	0.750	0.498	0.917
ph	0.781	0.784	0.781	0.830
pi	0.722	0.698	0.712	0.863
po	0.905	0.905	0.905	0.939
r	0.733	0.728	0.733	0.773
sb	0.714	0.719	0.712	0.742
se	0.931	0.930	0.936	0.937
te	0.892	0.889	0.894	0.917
th	0.985	0.980	0.985	0.987
ti	0.777	0.762	0.734	0.701
tw	0.723	0.722	0.723	0.789
wd	0.901	0.891	0.901	0.965
wi	0.958	0.954	0.953	0.960
wr	0.829	0.844	0.819	0.896
ww	0.835	0.838	0.838	0.873
y	0.863	0.864	0.863	0.912
rank	2.85	2.83	3.04	1.28

Table 4.19: F-score of the majority voting, random forest, the proposed algorithm and the improved one along with Friedman ranks.

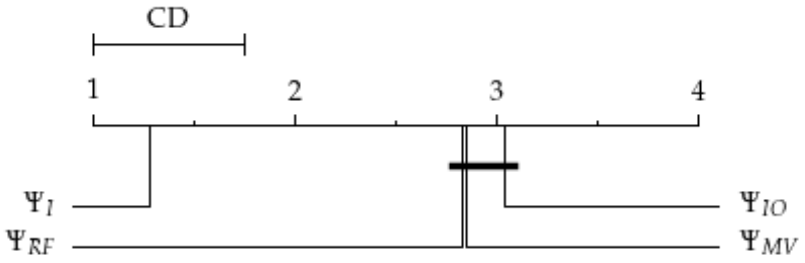
	$\mathrm{F-score}_{\mu}$				$\mathrm{F-score}_{M}$			
Dataset	Ψ_{MV}	Ψ_{RF}	Ψ_{IO}	Ψ_{I}	Ψ_{MV}	Ψ_{RF}	Ψ_{IO}	Ψ_{I}
ap	0.894	0.777	0.842	0.950	0.891	0.651	0.779	0.937
ba	0.692	0.685	0.693	0.711	0.479	0.473	0.483	0.498
bi	0.723	0.723	0.661	0.720	0.698	0.702	0.461	0.638
bu	0.544	0.546	0.558	0.819	0.523	0.535	0.530	0.815
c	0.839	0.879	0.839	0.898	0.856	0.873	0.856	0.891
d	0.924	0.942	0.925	0.969	0.924	0.942	0.926	0.969
e	0.410	0.413	0.410	0.427	0.119	0.121	0.119	0.226
h	0.729	0.719	0.760	0.849	0.575	0.520	0.624	0.800
io	0.808	0.830	0.762	0.866	0.797	0.824	0.751	0.853
ir	0.960	0.985	0.960	0.980	0.963	0.986	0.963	0.980
ma	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
me	0.534	0.750	0.498	0.918	0.549	0.745	0.533	0.912
ph	0.781	0.784	0.781	0.831	0.721	0.734	0.722	0.796
pi	0.722	0.698	0.712	0.864	0.687	0.659	0.674	0.852
po	0.905	0.905	0.905	0.938	0.475	0.475	0.475	0.732
r	0.733	0.728	0.733	0.770	0.741	0.735	0.742	0.777
sb	0.714	0.719	0.712	0.744	0.684	0.690	0.682	0.726
se	0.931	0.930	0.936	0.954	0.482	0.533	0.533	0.655
te	0.408	0.387	0.416	0.545	0.406	0.381	0.415	0.552
th	0.977	0.969	0.978	0.983	0.863	0.807	0.871	0.894
ti	0.777	0.762	0.734	0.741	0.744	0.721	0.702	0.442
tw	0.723	0.722	0.723	0.789	0.724	0.723	0.723	0.789
wd	0.901	0.891	0.901	0.965	0.901	0.890	0.902	0.963
wi	0.958	0.954	0.953	0.961	0.952	0.947	0.948	0.956
wr	0.486	0.533	0.457	0.689	0.230	0.240	0.217	0.497
ww	0.465	0.476	0.475	0.572	0.250	0.250	0.232	0.390
y	0.382	0.386	0.382	0.557	0.224	0.263	0.214	0.589
rank	2.89	2.81	3.06	1.24	2.91	2.74	3.07	1.28

Table 4.20: Micro–average precision and recall of the majority voting, random forest, the proposed algorithm and the improved one along with Friedman ranks.

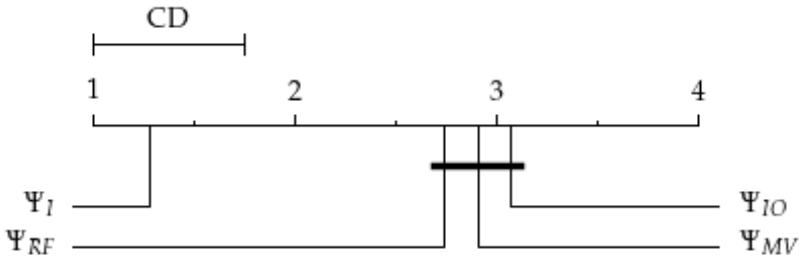
	Precision_μ				Recall_μ			
Dataset	Ψ_{MV}	Ψ_{RF}	Ψ_{IO}	Ψ_I	Ψ_{MV}	Ψ_{RF}	Ψ_{IO}	Ψ_I
ap	0.894	0.777	0.842	0.950	0.894	0.777	0.842	0.950
ba	0.692	0.685	0.693	0.708	0.692	0.685	0.693	0.713
bi	0.723	0.723	0.661	0.719	0.723	0.723	0.661	0.721
bu	0.544	0.546	0.558	0.824	0.544	0.546	0.558	0.815
c	0.839	0.879	0.839	0.898	0.839	0.879	0.839	0.898
d	0.924	0.942	0.925	0.969	0.924	0.942	0.925	0.969
e	0.410	0.417	0.410	0.427	0.410	0.413	0.410	0.427
h	0.729	0.719	0.760	0.845	0.729	0.719	0.760	0.853
io	0.808	0.830	0.762	0.863	0.808	0.830	0.762	0.869
ir	0.960	0.985	0.960	0.980	0.960	0.985	0.960	0.980
ma	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
me	0.534	0.750	0.498	0.918	0.534	0.750	0.498	0.918
ph	0.781	0.784	0.781	0.832	0.781	0.784	0.781	0.830
pi	0.722	0.698	0.712	0.865	0.722	0.698	0.712	0.862
po	0.905	0.905	0.905	0.938	0.905	0.905	0.905	0.938
r	0.733	0.728	0.733	0.778	0.733	0.728	0.733	0.762
sb	0.714	0.719	0.712	0.742	0.714	0.719	0.712	0.747
se	0.931	0.930	0.936	0.941	0.931	0.930	0.936	0.967
te	0.408	0.387	0.416	0.542	0.408	0.387	0.416	0.548
th	0.977	0.969	0.978	0.984	0.977	0.969	0.978	0.982
ti	0.777	0.762	0.734	0.747	0.777	0.762	0.734	0.734
tw	0.723	0.722	0.723	0.788	0.723	0.722	0.723	0.790
wd	0.901	0.891	0.901	0.965	0.901	0.891	0.901	0.965
wi	0.958	0.954	0.953	0.961	0.958	0.954	0.953	0.961
wr	0.486	0.533	0.457	0.689	0.486	0.533	0.457	0.689
ww	0.465	0.476	0.475	0.565	0.465	0.476	0.475	0.579
y	0.382	0.386	0.382	0.559	0.382	0.386	0.382	0.555
rank	2.89	2.81	3.06	1.24	2.89	2.81	3.04	1.26

Table 4.21: Macro–average precision and recall of the majority voting, random forest, the proposed algorithm and the improved one along with Friedman ranks.

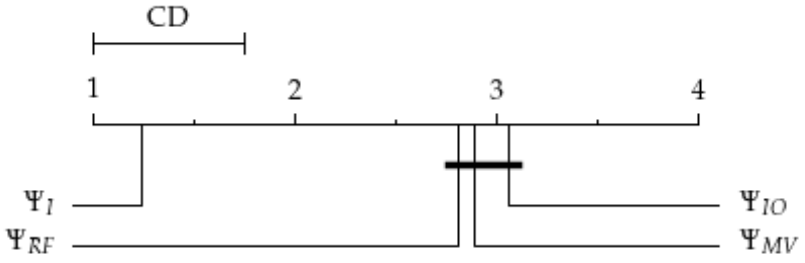
	Precision_M				Recall_M			
Dataset	Ψ_{MV}	Ψ_{RF}	Ψ_{IO}	Ψ_{I}	Ψ_{MV}	Ψ_{RF}	Ψ_{IO}	Ψ_{I}
ap	0.938	0.661	0.792	0.949	0.854	0.652	0.781	0.925
ba	0.460	0.453	0.464	0.477	0.500	0.496	0.504	0.521
bi	0.697	0.700	0.410	0.616	0.698	0.705	0.541	0.662
bu	0.529	0.535	0.543	0.820	0.517	0.535	0.518	0.811
c	0.826	0.833	0.826	0.891	0.895	0.923	0.895	0.891
d	0.927	0.942	0.928	0.969	0.922	0.942	0.924	0.969
e	0.085	0.087	0.085	0.206	0.204	0.204	0.204	0.250
h	0.600	0.521	0.679	0.847	0.553	0.525	0.581	0.758
io	0.801	0.837	0.747	0.872	0.794	0.811	0.756	0.834
ir	0.961	0.983	0.961	0.980	0.966	0.988	0.966	0.980
ma	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
me	0.536	0.749	0.527	0.910	0.577	0.741	0.546	0.913
ph	0.728	0.732	0.728	0.794	0.714	0.736	0.716	0.797
pi	0.687	0.661	0.675	0.851	0.688	0.657	0.673	0.854
po	0.452	0.452	0.452	0.818	0.500	0.500	0.500	0.662
r	0.752	0.744	0.752	0.795	0.731	0.726	0.732	0.760
sb	0.718	0.722	0.714	0.758	0.654	0.660	0.652	0.697
se	0.467	0.593	0.593	0.824	0.498	0.507	0.505	0.544
te	0.400	0.372	0.410	0.556	0.412	0.391	0.420	0.548
th	0.840	0.809	0.856	0.883	0.886	0.805	0.886	0.905
ti	0.835	0.777	0.844	0.373	0.671	0.675	0.601	0.542
tw	0.724	0.723	0.724	0.788	0.723	0.722	0.723	0.790
wd	0.903	0.890	0.904	0.963	0.899	0.890	0.900	0.963
wi	0.953	0.949	0.941	0.959	0.952	0.945	0.956	0.953
wr	0.237	0.237	0.220	0.603	0.223	0.243	0.214	0.423
ww	0.329	0.288	0.270	0.512	0.205	0.224	0.206	0.315
y	0.212	0.261	0.200	0.679	0.239	0.266	0.236	0.520
rank	2.89	2.83	3.00	1.28	2.98	2.69	2.91	1.43



(a) CD graph of Friedman ranks with Bonferroni–Dunn critical value for average accuracy and 9 base classifiers.



(b) CD graph of Friedman ranks with Bonferroni–Dunn critical value for $\mathrm{F} - \mathrm{score}_M$ and 9 base classifiers.



(c) CD graph of Friedman ranks with Bonferroni–Dunn critical value for $\mathrm{F} - \mathrm{score}_\mu$ and 9 base classifiers.

Figure 4.17: Friedman ranks comparison of the quality of the proposed algorithm, majority voting and random forest.

4.4.3. Conclusions

This work presents a new approach to the problem of clustering and selection, significant from determining MCS point of view. The feature space is divided into subregions based on the decision boundaries defined by the base classifier models. The selection process does not require concerning base classifiers, but the previously determined subspaces of the feature space. In a subspace of the feature space, the centroid of the objects belonging to it is calculated. The centroids of the selected subspaces of the feature space form a definition between the regions and are used in the weighted majority voting rule to define the final MCS decision. Single class label is assigned to all objects belonging to a specific subspace of the feature space.

The experimental results show that the proposed method obtains better classification results than the referential techniques and the method proposed earlier. The obtained outcome indicates an improvement in the quality of the classification, which is statistically significant. Such results were obtained for seven different performance classification measures.

The experiments described here were conducted using decision trees, which generate the decision boundaries necessary for the proposed integration process. Based on the obtained promising results, future research will focus, among other things, on the use of other base classifiers or the dynamic selection of the feature space subregions. In the proposal described in the article, the selection is static and depends on the imbalance ratio of the minority class in a given subregion. The source code of the evaluating program is hosted on github: https://github.com/TAndronicus/dynamic-ring.

Chapter 5

Conclusions

The research in the field of machine learning is very dynamic. One of the most exploited branches both in science and industry is ensemble learning, which involves creating a pool of base classifiers. Using the geometric representation has proven itself to provide promising results, although the approach itself is not as popular as, for example, classifier stacking.

The research problem presented in the introduction states:

> **Research hypothesis.** *Utilization of trained decision trees' decision boundaries allows for building an ensemble of classifiers with a greater value of performance quality measure than that of the random forest or majority voting using the same set of trained decision trees.*

The works presented confirm the hypothesis. Multiple algorithms were described, implemented and evaluated using the open–source benchmarking datasets from UCI and KEEL platforms as described in the chapter 3 to confirm the statement. Diverse classification performance measures were computed.

The first work describes an integration algorithm employing two distances in the feature space with the classification boundaries of trained classifiers: from the centroid and from the decision boundary. The label for the object under test is assigned according to the model prediction. The centroid with the corresponding label is taken to calculate the distance. Both distance values were mapped using the Gaussian function and different parameters of the mapping values were examined. The additional mapping serves two purposes:

- The distances are normalized to the values in the range of $[0, 1]$.

- The calculated value intuitively corresponds to the negative impact of too small or large distance from the decision boundary.

The final classification is done using the linear combination of the contributions calculated for both distances. Several possible distributions were evaluated to find the model with the highest quality measure. The statistical tests showed the significant improvement in classification performance when using the presented technique in comparison to the referential ensemble methods.

The second proposal is an algorithm that utilizes the static space division to produce the ensemble model. The feature space is divided into equal subspaces whose label is determined by the midpoint classification. Together with the label, the weight is calculated for the subspaces based on the volume of classification regions. Classification regions are the cuboids of the highest volume possible spanning over the points assigned a single label by the trained decision tree. Additional theoretical work presents a formal proof that the presented algorithm is a generalization of the (weighted) majority voting. The proposal was also implemented and examined on the datasets to expose the statistically significant classification quality improvement.

For the completeness of the results according to the experimental setup (see chapter 3), additional experiments were conducted for the algorithms presented in the two first articles. The extended results can be found in appendixes A and B.

In the third algorithm design, the approach was changed to dynamic space partitioning. The division of the feature space is based on the geometric representation of the trained model. Every competence region is assigned a label based on the classification of the region itself and its neighbors. Half of the weight is assigned by the subspace itself and the weights of the neighbors sum up to the other half. The contributions depend on the distance between the middle points of the subspaces. Similarly as in the previous works, statistical tests were conducted and the performance improvement against the referential majority voting and random forest was found.

The last work presented is similar to the previous in the sense that dynamic division of feature space is applied and the subspaces have an impact on their neighbors' classification. Several novelties were introduced:

- The average of the training points is used instead of midpoints to define the distance between the neighbors.
- To improve the effectiveness of the presented method for imbalanced dataset, additional filtering based on the training points distribution is applied.
- A wider range of neighbor subspaces can impact the weight calculation.
- Bagging is utilized in the evaluation process.

The new technique is examined using the datasets against both referential methods (majority voting and random forest) and the previously described. Statistically significant enhancement is observed against all the algorithms mentioned.

The works follow a consistent experimental setup presented in the chapter 3. Multiple datasets are used in the evaluation for increased reliability. Several classification quality measures are calculated because of their diverse intent: ACC, MCC and F-score for binary and micro– and macro–averaged precision, recall and F-score for other problems. All of the presented algorithms are implemented in Scala using Spark. For reproducibility the code is hosted on github and is publicly accessible.

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

References

[1] Keel data–mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Multiple-Valued Logic and Soft Computing*, 17(2 – 3):255 – 287, 2011.

[2] Integration of a deep learning classifier with a random forest approach for predicting malonylation sites. *Genomics, Proteomics Bioinformatics*, 16(6):451 – 459, 2018.

[3] J. Abellán and J. G. Castellano. A comparative study on base classifiers in ensemble methods for credit scoring. *Expert Systems with Applications*, 73:1 – 10, 2017.

[4] P. S. Akash, M. E. Kadir, A. A. Ali, M. N. Ahad Tawhid, and M. Shoyaib. Introducing confidence as a weight in random forest. In *2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST)*, pages 611 – 616, 2019.

[5] E. Alpaydin. *Introduction to Machine Learning*. 2004.

[6] E. Alpaydin and M. I. Jordan. Local linear perceptrons for classification. *IEEE Transactions on Neural Networks*, 7(3):788 – 794, 1996.

[7] F. Alzami, J. Tang, Z. Yu, S. Wu, C. P. Chen, J. You, and J. Zhang. Adaptive hybrid feature selection-based classifier ensemble for epileptic seizure classification. *IEEE access*, 6:29132 – 29145, 2018.

[8] G. Armano and E. Tamponi. Building forests of local trees. *Pattern Recognition*, 76:380 – 390, 2018.

[9] M. Asafuddoula, B. Verma, and M. Zhang. A divide-and-conquer-based ensemble classifier learning by means of many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 22(5):762 – 777, 2018.

[10] S. Bashir, U. Qamar, and F. H. Khan. Intellihealth: a medical decision support application using a novel weighted multi–layer classifier ensemble framework. *Journal of biomedical informatics*, 59:185 – 200, 2016.

[11] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland. Data mining for credit card fraud: A comparative study. *decision support systems*, 50(3):602 – 613, 2011.

[12] G. Biau and L. Devroye. *Lectures on the nearest neighbor method*. Springer, 2015.

[13] J. Biedrzycki. Decision tree integration algorithm using static regions of competence and geometric representation. *Polskie Porozumienie na Rzecz Rozwoju Sztucznej Inteligencji: 16–18.10.2019, Wrocław, Poland : conference proceedings*, pages 29 – 32, 2019.

[14] J. Biedrzycki and R. Burduk. Decision tree integration using dynamic regions of competence. *Entropy*, 22(10):1129, 2020.

[15] J. Biedrzycki and R. Burduk. Integration of decision trees using distance to centroid and to decision boundary. *Journal of Universal Computer Science*, 26:720 – 733, 2020.

[16] J. Biedrzycki and R. Burduk. Weighted scoring in geometric space for decision tree ensemble. *IEEE Access*, 8:82100 – 82107, 2020.

[17] C. M. Bishop. *Pattern Recognition and Machine Learning*. 2006.

[18] L. Breiman. *Classification and regression trees*. 1983.

[19] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123 – 140, 1996.

[20] L. Breiman. Random forests. *Machine Learning archive*, 45(1):5 – 32, 2001.

[21] A. S. Britto Jr, R. Sabourin, and L. E. Oliveira. Dynamic selection of classifiers – a comprehensive review. *Pattern recognition*, 47(11):3665 – 3680, 2014.

[22] R. Burduk. Classifier fusion with interval–valued weights. *Pattern Recognition Letters*, 34(14):1623 – 1629, 2013.

[23] R. Burduk. Integration base classifiers in geometry space by harmonic mean. In L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, and J. M. Zurada, editors, *Artificial Intelligence and Soft Computing*, pages 585 – 592, Cham, 2018. Springer International Publishing.

[24] R. Burduk and J. Biedrzycki. Integration and selection of linear SVM classifiers in geometric space. *Journal of Universal Computer Science*, 25(6):718 – 730, Jun 2019.

[25] R. Burduk and J. Biedrzycki. Integration of linear SVM classifiers in geometric space using the median. In M. Choraś and R. S. Choraś, editors, *Image Processing and Communications Challenges 10*, pages 30 – 36, Cham, 2019. Springer International Publishing.

[26] R. Burduk, W. Bożejko, and S. Zacher. Novel approach to gentle AdaBoost algorithm with linear weak classifiers. In N. T. Nguyen, K. Jearanaitanakij, A. Selamat, B. Trawiński, and S. Chittayasothorn, editors, *Intelligent Information and Database Systems*, pages 600 – 611, Cham, 2020. Springer International Publishing.

[27] R. Burduk and A. Kasprzak. The use of geometric mean in the process of integration of three base classifiers. In *IFIP International Conference on Computer Information Systems and Industrial Management*, pages 246 – 253, 2018.

[28] C. Cai, D. K. Wornyo, L. Wang, and X. Shen. Building weighted classifier ensembles through classifiers pruning. In B. Huet, L. Nie, and R. Hong, editors, *Internet Multimedia Computing and Service*, pages 131 – 139, Singapore, 2018. Springer Singapore.

[29] J. Cao, G. Lv, C. Chang, and H. Li. A feature selection based serial SVM ensemble classifier. *IEEE Access*, 7:144516–144523, 2019.

[30] Z. Cao, X. Pan, Y. Yang, Y. Huang, and H.-B. Shen. The lnclocator: a subcellular localization predictor for long non-coding RNAs based on a stacked ensemble classifier. *Bioinformatics*, 34(13):2185 – 2194, 2018.

[31] P. R. Cavalin, R. Sabourin, and C. Y. Suen. Dynamic selection approaches for multiple classifier systems. *Neural computing and applications*, 22(3-4):673 – 688, 2013.

[32] N. V. Chawla, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. Learning ensembles from bites: A scalable and accurate approach. *J. Mach. Learn. Res.*, 5:421 – 451, Dec. 2004.

[33] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785 – 794, 2016.

[34] R. Das, I. Turkoglu, and A. Sengur. Effective diagnosis of heart disease through neural networks ensembles. *Expert Systems With Applications*, 36(4):7675 – 7680, 2009.

[35] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1 – 30, Dec. 2006.

[36] M. Ding, S. Antani, S. Jaeger, Z. Xue, S. Candemir, M. Kohli, and G. Thoma. Local–global classifier fusion for screening chest radiographs. In T. S. Cook and J. Zhang, editors, *Medical Imaging 2017: Imaging Informatics for Healthcare, Research, and Applications*, volume 10138, pages 64 – 69. International Society for Optics and Photonics, SPIE, 2017.

[37] F. Dong, Q. Li, D. Xu, W. Xiu, Q. Zeng, X. Zhu, F. Xu, B. Jiang, and M. Zhang. Differentiation between pilocytic astrocytoma and glioblastoma: a decision tree model using contrast-enhanced magnetic resonance imaging-derived quantitative radiomic features. *European Radiology*, 29(8):3968 – 3975, 2019.

[38] D. Dua and C. Graff. UCI machine learning repository, 2017.

[39] J.-H. Eom, S.-C. Kim, and B.-T. Zhang. Aptacdss-e: A classifier ensemble-based clinical decision support system for cardiovascular disease level prediction. *Expert Systems With Applications*, 34(4):2465 – 2479, 2008.

[40] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The journal of machine learning research*, 15(1):3133 – 3181, 2014.

[41] J. Fierrez, A. Morales, R. Vera-Rodriguez, and D. Camacho. Multiple classifiers in biometrics. part 1: Fundamentals and review. *Information Fusion*, 44:57 – 64, 2018.

[42] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *conference on learning theory*, 55(1):119 – 139, 1997.

[43] Y. Freund and R. E. Schapire. A short introduction to boosting. 1999.

[44] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189 – 1232, 2001.

[45] G. Giacinto and F. Roli. An approach to the automatic design of multiple classifier systems. *machine learning and data mining in pattern recognition*, 22(1):25 – 33, 2001.

[46] G. Giacinto and F. Roli. Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing*, 19(9):699 – 707, 2001.

[47] N. Gong, Z. Wang, S. Chen, G. Liu, and S. Xin. Decision boundary extraction of classifiers. *Journal of Physics: Conference Series*, 1651:012031, nov 2020.

[48] J. Gou, Y. Zhan, Y. Rao, X. Shen, X. Wang, and W. He. Improved pseudo nearest neighbor classification. *Knowledge Based Systems*, 70(70):361 – 375, 2014.

[49] S. Guo, H. He, and X. Huang. A multi–stage self–adaptive classifier ensemble model with application in credit scoring. *IEEE Access*, 7:78549 – 78559, 2019.

[50] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[51] M. V. Gwetu, S. Viriri, and J.-R. Tapamo. Purity and out of bag confidence metrics for random forest weighting. In N. T. Nguyen, E. Pimenidis, Z. Khan, and B. Trawiński, editors, *Computational Collective Intelligence*, pages 491 – 502, Cham, 2018. Springer International Publishing.

[52] A. Hajdu, L. Hajdu, A. Jónás, L. Kovács, and H. Tomán. Generalizing the majority voting scheme to spatially constrained voting. *IEEE Transactions on Image Processing*, 22(11):4182 – 4194, Nov 2013.

[53] T. Hastie, R. J. Tibshirani, and J. Friedman. The elements of statistical learning : data mining, inference,and prediction. *The Mathematical Intelligencer*, 27(2):83 – 85, 2005.

[54] S. Haykin. *Neural Networks: A Comprehensive Foundation (3rd Edition)*. Prentice-Hall, Inc., USA, 2007.

[55] F. Hou, L. Wentai, H. Li, J. Ren, G. Liu, and Q. Gu. Impr-co-forest: The improved co-forest algorithm based on optimized decision tree and dual-confidence estimation method. 2019.

[56] Y. Huang, N. He, Y. Chen, Z. Chen, and L. Li. BERMP: a cross-species classifier for predicting m 6 a sites by integrating a deep learning algorithm and a random forest approach. *International Journal of Biological Sciences*, 14:1669 – 1677, 09 2018.

[57] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90 – 95, 2007.

[58] K. Ihou, N. Bouguila, and W. Bouachir. Efficient integration of generative topic models into discriminative classifiers using robust probabilistic kernels. *Pattern Analysis and Applications*, pages 1 – 25, 09 2020.

[59] I. Islek and S. G. Ögüdücü. A decision support system for demand forecasting based on classifier ensemble. In *FedCSIS (Communication Papers)*, pages 35 – 41, 2017.

[60] K. Jackowski, D. Jankowski, P. Ksieniewicz, D. Simić, S. Simić, and M. Woźniak. Ensemble classifier systems for headache diagnosis. pages 273 – 284, 2014.

[61] M. Javadi, R. Ebrahimpour, A. Sajedin, S. Faridi, and S. Zakernejad. Improving ECG classification accuracy using an ensemble of neural network modules. *PLoS one*, 6(10):e24386, 2011.

[62] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001.

[63] C. Ju, A. Bibaut, and M. J. van der Laan. The relative performance of ensemble methods with deep convolutional neural networks for image classification. *Journal of Applied Statistics*, 45(15):2800 – 2818, 2018.

[64] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. LightGBM: a highly efficient gradient boosting decision tree. In *NIPS'17 Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3149 – 3157, 2017.

[65] S. R. Kheradpisheh, F. Behjati-Ardakani, and R. Ebrahimpour. Combining classifiers using nearest decision prototypes. *Applied Soft Computing*, 13(12):4570 – 4578, 2013.

[66] E. Kim and J. Ko. Dynamic classifier integration method. In N. C. Oza, R. Polikar, J. Kittler, and F. Roli, editors, *Multiple Classifier Systems*, pages 97 – 107, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[67] Y. Kim and S. Y. Sohn. Stock fraud detection using peer group analysis. *Expert Systems With Applications*, 39(10):8986 – 8992, 2012.

[68] J. Kittler, M. Hatef, R. P. Duin, and J. Matas. On combining classifiers. *IEEE transactions on pattern analysis and machine intelligence*, 20(3):226 – 239, 1998.

[69] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica (lithuanian Academy of Sciences)*, 31(3):249 – 268, 2007.

[70] B. Krawczyk and B. Cyganek. Selecting locally specialised classifiers for one–class classification ensembles. *Pattern analysis and applications*, 20(2):427 – 439, 2017.

[71] P. Ksieniewicz and R. Burduk. Clustering and weighted scoring in geometric space support vector machine ensemble for highly imbalanced data classification. In *International Conference on Computational Science*, pages 128 – 140. Springer, 2020.

[72] L. Kuncheva. A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):281 – 286, 2002.

[73] L. I. Kuncheva. *Combining Pattern Classifiers*. 2004.

[74] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing*, 30(2):457 – 474, 2000.

[75] D. T. Larose and C. D. Larose. Discovering knowledge in data: an introduction data mining. ed. 2. 2019.

[76] Y. Li, L.-P. Li, L. Wang, C.-Q. Yu, Z. Wang, and Z.-H. You. An ensemble classifier to predict protein–protein interactions by combining pssm-based evolutionary information with local binary pattern model. *International Journal of Molecular Sciences*, 20:3511, 07 2019.

[77] D. Liang, C.-F. Tsai, A.-J. Dai, and W. Eberle. A novel classifier ensemble approach for financial distress prediction. *Knowledge and Information Systems*, 54(2):437 – 462, 2018.

[78] B. Liu, F. Yang, and K.-C. Chou. 2L-piRNA: A two–layer ensemble classifier for identifying piwi-interacting RNAs and their function. *Molecular therapy. Nucleic acids*, 7:267 – 277, 2017.

[79] P. Luo, H. Xiong, K. Lü, and Z. Shi. Distributed classification in peer–to–peer networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 968 – 976, New York, NY, USA, 2007. ACM.

[80] P. López García, A. Masegosa, E. Osaba, E. Onieva, and A. Perallos. Ensemble classification for imbalanced data based on feature space partitioning and hybrid metaheuristics. *Applied Intelligence*, 49, 08 2019.

[81] S. Mao, L. Jiao, L. Xiong, S. Gou, B. Chen, and S.-K. Yeung. Weighted classifier ensemble based on quadratic form. *Pattern Recognition*, 48(5):1688 – 1706, 2015.

[82] D. D. Margineantu and T. G. Dietterich. Pruning adaptive boosting. In *ICML '97 Proceedings of the Fourteenth International Conference on Machine Learning*, pages 211 – 218, 1997.

[83] A. B. R. McIntyre, R. Ounit, E. Afshinnekoo, R. J. Prill, E. Hénaff, N. Alexander, S. S. Minot, D. Danko, J. Foox, S. Ahsanuddin, S. Tighe, N. A. Hasan, P. Subramanian, K. Moffat, S. Levy, S. Lonardi, N. Greenfield, R. R. Colwell, G. L. Rosen, and C. E. Mason. Comprehensive benchmarking and ensemble approaches for metagenomic classifiers. *Genome Biology*, 18(1):182 – 182, 2017.

[84] W. McKinney. Data structures for statistical computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference, Austin, TX, USA*, pages 56 – 61, 2010.

[85] M. A. Medina-Pérez, R. Monroy, J. B. Camiña, and M. García-Borroto. Bagging-TPMiner: A classifier ensemble for masquerader detection based on typical objects. *Soft Computing*, 21(3):557 – 569, 2017.

[86] J. Mendes-Moreira, C. Soares, A. M. Jorge, and J. F. D. Sousa. Ensemble approaches for regression: A survey. *Acm computing surveys (csur)*, 45(1):1 – 40, 2012.

[87] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar. MLlib: Machine learning in Apache Spark, 2015.

[88] S. T. Miller and C. Busby-Earle. Multi–perspective machine learning a classifier ensemble method for intrusion detection. In *Proceedings of the 2017 International Conference on Machine Learning and Soft Computing*, pages 7 – 12, 2017.

[89] M. Mohandes, M. Deriche, and S. O. Aliyu. Classifiers combination techniques: A comprehensive review. *IEEE Access*, 6:19626 – 19639, 2018.

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

[91] T. T. Nguyen, M. P. Nguyen, X. C. Pham, A. W.-C. Liew, and W. Pedrycz. Combining heterogeneous classifiers via granular prototypes. *Applied Soft Computing*, 73:795 – 815, 2018.

[92] T. Oliphant. NumPy: A guide to NumPy. USA: Trelgol Publishing, 2006.

[93] J. Ortigosa-Hernández, I. Inza, and J. A. Lozano. Measuring the class–imbalance extent of multi–class problems. *Pattern Recognition Letters*, 98:32 – 38, 2017.

[94] M. G. Padalkar, C. Beltrán-González, M. Bustreo, A. Del Bue, and V. Murino. A versatile crack inspection portable system based on classifier ensemble and controlled illumination. *arXiv preprint arXiv:2010.09557*, 2020.

[95] B. T. Pham, D. T. Bui, M. Dholakia, I. Prakash, H. V. Pham, K. Mehmood, and H. Q. Le. A novel ensemble classifier of rotation forest and naïve bayer for landslide susceptibility assessment at the luc yen district, yen bai province (viet nam) using gis. *Geomatics, Natural Hazards and Risk*, 8(2):649 – 671, 2017.

[96] B. T. Pham, D. T. Bui, I. Prakash, and M. Dholakia. Rotation forest fuzzy rule-based classifier ensemble for spatial prediction of landslides using GIS. *Natural Hazards*, 83(1):97 – 127, 2016.

[97] C. O. Plumpton, L. I. Kuncheva, N. N. Oosterhof, and S. J. Johnston. Naive random subspace ensemble with linear classifiers for real–time classification of fMRI data. *Pattern Recognition*, 45(6):2101 – 2108, 2012.

[98] V. Polianskii and F. T. Pokorny. Voronoi boundary classification: A high–dimensional geometric approach via weighted Monte Carlo integration. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5162 – 5170, Long Beach, California, USA, Jun 2019. PMLR.

[99] M. P. Ponti. Combining classifiers: From the creation of ensembles to the decision fusion. In *2011 24th SIBGRAPI Conference on Graphics, Patterns, and Images Tutorials*, pages 1 – 10, 2011.

[100] C. Potes, S. Parvaneh, A. Rahman, and B. Conroy. Ensemble of feature–based and deep learning–based classifiers for detection of abnormal heart sounds. In *2016 Computing in Cardiology Conference (CinC)*, pages 621 – 624, 2016.

[101] M. Przybyla-Kasperek and A. Wakulicz-Deja. Comparison of fusion methods from the abstract level and the rank level in a dispersed decision–making system. *International Journal of General Systems*, 46(4):386 – 413, 2017.

[102] O. Pujol and D. Masip. Geometry-based ensembles: Toward a structural characterization of the classification boundary. *IEEE transactions on pattern analysis and machine intelligence*, 31:1140 – 6, 07 2009.

[103] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81 – 106, 1986.

[104] J. R. Quinlan. *C4.5: Programs for Machine Learning*. 1992.

[105] A. F. R. Rahman, H. Alam, and M. C. Fairhurst. Multiple classifier combination for character recognition: Revisiting the majority voting system and its variations. In D. Lopresti, J. Hu, and R. Kashi, editors, *Document Analysis Systems V*, pages 167 – 178, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[106] I. Rejer. Genetic algorithms for feature selection for brain-computer interface. *IJPRAI*, 29, 2015.

[107] L. Rokach. Decision forest: Twenty years of research. *Information Fusion*, 27:111 – 125, 2016.

[108] L. Rokach and O. Maimon. Top–down induction of decision trees classifiers – a survey. *systems man and cybernetics*, 35(4):476 – 487, 2005.

[109] M. Sabzevari, G. Martínez-Muñoz, and A. Suárez. Vote–boosting ensembles. *Pattern Recognition*, 83:119 – 133, 2018.

[110] O. Sagi and L. Rokach. Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[111] F. Salo, A. B. Nassif, and A. Essex. Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection. *Computer Networks*, 148:164 – 175, 2019.

[112] M. Saqlain, B. Jargalsaikhan, and J. Y. Lee. A voting ensemble classifier for wafer map defect patterns identification in semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing*, 32(2):171 – 182, 2019.

[113] R. E. Schapire. The strength of weak learnability. *Machine learning*, 5(2):197 – 227, 1990.

[114] J. Semerdjian and S. Frank. An ensemble classifier for predicting the onset of type ii diabetes, 2017.

[115] A. Shabtai, R. Moskovitch, Y. Elovici, and C. Glezer. Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey. *Information Security Technical Report*, 14(1):16 – 29, 2009.

[116] A. Shashua. Introduction to machine learning. 2009.

[117] R. P. Sheridan, W. M. Wang, A. Liaw, J. Ma, and E. M. Gifford. Extreme gradient boosting as a method for quantitative structure–activity relationships. *Journal of Chemical Information and Modeling*, 56(12):2353 – 2360, 2016.

[118] E. J. R. Silva and C. Zanchettin. A Voronoi diagram based classifier for multiclass imbalanced data sets. In *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 109 – 114, 2016.

[119] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(4):427 – 437, 2009.

[120] M. Sultan Zia, M. Hussain, and M. Arfan Jaffar. A novel spontaneous facial expression recognition using dynamically weighted majority voting based ensemble classifier. *Multimedia Tools and Applications*, 77(19):25537 – 25567, Oct 2018.

[121] S. B. Taieb and R. J. Hyndman. A gradient boosting approach to the kaggle load forecasting competition. *International Journal of Forecasting*, 30(2):382 – 394, 2014.

[122] B. A. Tama, A. S. Patil, and K.-H. Rhee. An improved model of anomaly detection using two–level classifier ensemble. In *2017 12th Asia Joint Conference on Information Security (AsiaJCIS)*, pages 1 – 4. IEEE, 2017.

[123] P.-N. Tan, M. M. Steinbach, and V. Kumar. *Introduction to Data Mining*. 2005.

[124] C. Tekin, J. Yoon, and M. van der Schaar. Adaptive ensemble learning with confidence bounds. *IEEE Transactions on Signal Processing*, 65(4):888–903, 2017.

[125] M. Termenon and M. Graña. A two stage sequential ensemble applied to the classification of Alzheimer's disease based on MRI features. *Neural Processing Letters*, 35(1):1 – 12, 2012.

[126] P. Trajdos and R. Burduk. Combination of linear classifiers using score function – analysis of possible combination strategies. In *International Conference on Computer Recognition Systems*, pages 348 – 359, 2019.

[127] P. Trajdos and R. Burduk. Combining linear classifiers using probability–based potential functions. *IEEE Access*, 8:207947–207961, 2020.

[128] B. Trawiński, T. Lasota, O. Kempa, Z. Telec, and M. Kutrzyński. Comparison of ensemble learning models with expert algorithms designed for a property valuation system. In *International Conference on Computational Collective Intelligence*, pages 317 – 327, 2017.

[129] V. Van Asch. Macro– and micro–averaged evaluation measures. *Belgium: CLiPS*, 49, 2013.

[130] L. Wang, Z.-H. You, S.-X. Xia, F. Liu, X. Chen, X. Yan, and Y. Zhou. Advancing the prediction accuracy of protein-protein interactions by utilizing evolutionary information from position-specific scoring matrix and ensemble classifier. *Journal of Theoretical Biology*, 418:105 – 110, 2017.

[131] Y. Wang, F. Liu, and A. Zhu. Bearing fault diagnosis based on a hybrid classifier ensemble approach and the improved Dempster–Shafer theory. *Sensors*, 19(9):2097, 2019.

[132] K. Woods, W. P. Kegelmeyer, and K. Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):405 – 410, 1997.

[133] M. Woźniak, M. Graña, and E. Corchado. A survey of multiple classifier systems as hybrid systems. *Information Fusion*, 16:3 – 17, 2014.

[134] L. Xu, A. Krzyzak, and C. Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE transactions on systems, man, and cybernetics*, 22(3):418 – 435, 1992.

Appendix A

Extended results for the first algorithm

This appendix contains extended results for the article presented in the section 4.1. It is divided regarding quality measures.

A.1. Accuracy

Table A.1: Extended ACC values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 5$ and $\gamma_\omega = 5$.

	Ψ_{MV}	Ψ_{RF}	$\Psi_{0.0}$	$\Psi_{0.3}$	$\Psi_{0.7}$	$\Psi_{1.0}$
aa	0.912	0.914	0.914	0.922	0.919	0.902
ap	0.852	0.821	0.802	0.825	0.887	0.859
ba	0.784	0.792	0.727	0.786	0.806	0.757
bi	0.723	0.721	0.724	0.727	0.735	0.728
bu	0.580	0.567	0.599	0.536	0.590	0.534
c	0.759	0.808	0.828	0.696	0.892	0.821
d	0.927	0.931	0.932	0.938	0.939	0.917
e	0.812	0.815	0.786	0.800	0.822	0.800
h	0.721	0.708	0.659	0.748	0.694	0.739
io	0.844	0.842	0.844	0.873	0.870	0.826
ir	0.966	0.965	0.973	0.974	0.952	0.946
me	0.669	0.667	0.697	0.592	0.660	0.630
po	0.916	0.915	0.832	0.924	0.901	0.934
ph	0.775	0.774	0.770	0.785	0.767	0.765
pi	0.736	0.735	0.737	0.768	0.733	0.767
ri	0.737	0.737	0.734	0.735	0.727	0.727
sb	0.707	0.712	0.694	0.697	0.708	0.704
se	0.936	0.935	0.939	0.938	0.934	0.931
tw	0.737	0.737	0.744	0.740	0.734	0.729
te	0.890	0.890	0.890	0.891	0.891	0.891
th	0.979	0.980	0.981	0.978	0.979	0.980
ti	0.781	0.774	0.780	0.787	0.782	0.786
wd	0.909	0.908	0.912	0.888	0.896	0.904
wi	0.939	0.940	0.934	0.936	0.926	0.946
wr	0.819	0.823	0.827	0.819	0.826	0.823
ww	0.837	0.839	0.840	0.831	0.839	0.833
y	0.865	0.868	0.868	0.871	0.861	0.862
rank	3.11	3.18	3.04	2.82	3.04	3.61

Table A.2: Extended ACC values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 20$ and $\gamma_\omega = 5$.

	Ψ_{MV}	Ψ_{RF}	$\Psi_{0.0}$	$\Psi_{0.3}$	$\Psi_{0.7}$	$\Psi_{1.0}$
aa	0.912	0.914	0.898	0.906	0.910	0.917
ap	0.852	0.821	0.842	0.865	0.897	0.870
ba	0.784	0.792	0.694	0.767	0.786	0.793
bi	0.723	0.721	0.737	0.721	0.774	0.731
bu	0.580	0.567	0.613	0.564	0.636	0.492
c	0.759	0.808	0.728	0.685	0.763	0.834
d	0.927	0.931	0.928	0.926	0.920	0.918
e	0.812	0.815	0.829	0.806	0.808	0.805
h	0.721	0.708	0.740	0.663	0.788	0.690
io	0.844	0.842	0.885	0.803	0.864	0.787
ir	0.966	0.965	0.977	0.990	0.994	0.958
me	0.669	0.667	0.610	0.714	0.798	0.657
po	0.916	0.915	0.785	0.919	0.904	0.903
ph	0.775	0.774	0.774	0.782	0.763	0.782
pi	0.736	0.735	0.743	0.736	0.760	0.737
ri	0.737	0.737	0.737	0.740	0.741	0.742
sb	0.707	0.712	0.718	0.704	0.715	0.707
se	0.936	0.935	0.943	0.938	0.933	0.941
tw	0.737	0.737	0.733	0.744	0.740	0.729
te	0.890	0.890	0.890	0.888	0.890	0.889
th	0.979	0.980	0.978	0.979	0.979	0.979
ti	0.781	0.774	0.777	0.773	0.771	0.802
wd	0.909	0.908	0.870	0.908	0.911	0.919
wi	0.939	0.940	0.921	0.933	0.938	0.954
wr	0.819	0.823	0.814	0.830	0.818	0.831
ww	0.837	0.839	0.846	0.834	0.844	0.839
y	0.865	0.868	0.853	0.866	0.872	0.864
rank	3.14	3.14	3.36	3.61	2.57	3.07

Table A.3: Extended ACC values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 20$ and $\gamma_\omega = 20$.

	Ψ_{MV}	Ψ_{RF}	$\Psi_{0.0}$	$\Psi_{0.3}$	$\Psi_{0.7}$	$\Psi_{1.0}$
aa	0.912	0.914	0.904	0.909	0.894	0.912
ap	0.852	0.821	0.628	0.907	0.846	0.877
ba	0.784	0.792	0.700	0.780	0.789	0.812
bi	0.723	0.721	0.721	0.726	0.700	0.741
bu	0.580	0.567	0.607	0.617	0.576	0.553
c	0.759	0.808	0.727	0.707	0.692	0.675
d	0.927	0.931	0.903	0.929	0.938	0.939
e	0.812	0.815	0.811	0.816	0.820	0.821
h	0.721	0.708	0.658	0.688	0.790	0.726
io	0.844	0.842	0.861	0.793	0.767	0.794
ir	0.966	0.965	0.960	0.982	0.981	0.973
me	0.669	0.667	0.598	0.543	0.660	0.683
po	0.916	0.915	0.687	0.925	0.891	0.929
ph	0.775	0.774	0.778	0.761	0.776	0.770
pi	0.736	0.735	0.707	0.761	0.713	0.731
ri	0.737	0.737	0.750	0.734	0.737	0.740
sb	0.707	0.712	0.707	0.715	0.707	0.716
se	0.936	0.935	0.928	0.932	0.946	0.933
tw	0.737	0.737	0.740	0.730	0.734	0.738
te	0.890	0.890	0.889	0.891	0.888	0.888
th	0.979	0.980	0.980	0.979	0.982	0.977
ti	0.781	0.774	0.774	0.783	0.765	0.785
wd	0.909	0.908	0.903	0.938	0.894	0.919
wi	0.939	0.940	0.953	0.931	0.935	0.947
wr	0.819	0.823	0.807	0.806	0.801	0.813
ww	0.837	0.839	0.831	0.843	0.844	0.846
y	0.865	0.868	0.853	0.866	0.862	0.873
rank	3.04	3.07	4	3.21	3.64	2.43

Table A.4: Extended ACC values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 10$ and $\gamma_\omega = 10$.

	Ψ_{MV}	Ψ_{RF}	$\Psi_{0.0}$	$\Psi_{0.3}$	$\Psi_{0.7}$	$\Psi_{1.0}$
aa	0.912	0.914	0.908	0.926	0.911	0.901
ap	0.852	0.821	0.752	0.832	0.849	0.838
ba	0.784	0.792	0.740	0.760	0.774	0.776
bi	0.723	0.721	0.724	0.710	0.729	0.702
bu	0.580	0.567	0.580	0.605	0.588	0.567
c	0.759	0.808	0.731	0.683	0.674	0.734
d	0.927	0.931	0.939	0.931	0.941	0.933
e	0.812	0.815	0.848	0.830	0.812	0.813
h	0.721	0.708	0.734	0.755	0.696	0.727
io	0.844	0.842	0.825	0.847	0.778	0.801
ir	0.966	0.965	0.959	0.976	0.964	0.956
me	0.669	0.667	0.703	0.683	0.818	0.644
po	0.916	0.915	0.741	0.922	0.924	0.934
ph	0.775	0.774	0.779	0.775	0.776	0.773
pi	0.736	0.735	0.756	0.731	0.750	0.731
ri	0.737	0.737	0.737	0.743	0.735	0.735
sb	0.707	0.712	0.718	0.699	0.708	0.690
se	0.936	0.935	0.932	0.923	0.937	0.928
tw	0.737	0.737	0.734	0.737	0.740	0.740
te	0.890	0.890	0.889	0.890	0.890	0.891
th	0.979	0.980	0.982	0.982	0.978	0.978
ti	0.781	0.774	0.788	0.764	0.763	0.783
wd	0.909	0.908	0.915	0.930	0.916	0.900
wi	0.939	0.940	0.948	0.937	0.939	0.948
wr	0.819	0.823	0.830	0.822	0.826	0.824
ww	0.837	0.839	0.841	0.841	0.828	0.830
y	0.865	0.868	0.864	0.867	0.87	0.871
rank	3.07	3.11	2.79	2.93	2.96	3.61

A.2. Microaveraged F-score

Table A.5: Extended microaveraged F-score values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 5$ and $\gamma_\omega = 5$.

	Ψ_{MV}	Ψ_{RF}	$\Psi_{0.0}$	$\Psi_{0.3}$	$\Psi_{0.7}$
aa	0.484	0.500	0.442	0.490	0.482
ap	0.856	0.824	0.819	0.825	0.887
ba	0.676	0.687	0.590	0.680	0.709
bi	0.723	0.721	0.724	0.727	0.735
bu	0.580	0.567	0.599	0.536	0.590
c	0.759	0.808	0.828	0.696	0.892
d	0.927	0.931	0.932	0.938	0.939
e	0.442	0.451	0.408	0.453	0.473
h	0.721	0.708	0.659	0.748	0.694
io	0.844	0.842	0.844	0.873	0.870
ir	0.950	0.948	0.959	0.961	0.928
me	0.644	0.668	0.697	0.592	0.660
po	0.916	0.915	0.832	0.924	0.901
ph	0.775	0.774	0.770	0.785	0.767
pi	0.736	0.735	0.737	0.768	0.733
ri	0.737	0.737	0.734	0.735	0.727
sb	0.707	0.712	0.694	0.697	0.708
se	0.936	0.935	0.939	0.938	0.934
tw	0.737	0.737	0.744	0.740	0.734
te	0.395	0.397	0.396	0.402	0.398
th	0.969	0.970	0.972	0.967	0.968
ti	0.781	0.774	0.780	0.787	0.782
wd	0.909	0.908	0.912	0.888	0.896
wi	0.939	0.940	0.934	0.936	0.926
wr	0.480	0.494	0.481	0.484	0.479
ww	0.464	0.469	0.462	0.460	0.457
y	0.359	0.372	0.359	0.372	0.360
rank	3.43	3.11	3.54	2.79	3.43

Table A.6: Extended microaveraged F-score values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 20$ and $\gamma_\omega = 5$.

	Ψ_{MV}	Ψ_{RF}	$\Psi_{0.0}$	$\Psi_{0.3}$	$\Psi_{0.7}$	$\Psi_{1.0}$
aa	0.484	0.500	0.485	0.462	0.472	0.504
ap	0.856	0.824	0.842	0.865	0.920	0.877
ba	0.676	0.687	0.541	0.650	0.679	0.689
bi	0.723	0.721	0.737	0.721	0.774	0.731
bu	0.580	0.567	0.613	0.564	0.636	0.492
c	0.759	0.808	0.728	0.685	0.763	0.834
d	0.927	0.931	0.928	0.926	0.920	0.918
e	0.442	0.451	0.472	0.422	0.503	0.387
h	0.721	0.708	0.740	0.663	0.788	0.69
io	0.844	0.842	0.885	0.803	0.864	0.787
ir	0.950	0.948	0.965	0.985	0.991	0.937
me	0.644	0.668	0.610	0.714	0.798	0.657
po	0.916	0.915	0.785	0.919	0.904	0.903
ph	0.775	0.774	0.774	0.782	0.763	0.782
pi	0.736	0.735	0.743	0.736	0.760	0.737
ri	0.737	0.737	0.737	0.740	0.741	0.742
sb	0.707	0.712	0.718	0.704	0.715	0.707
se	0.936	0.935	0.943	0.938	0.933	0.941
tw	0.737	0.737	0.733	0.744	0.740	0.729
te	0.395	0.397	0.396	0.385	0.396	0.392
th	0.969	0.970	0.967	0.969	0.969	0.968
ti	0.781	0.774	0.777	0.773	0.771	0.802
wd	0.909	0.908	0.870	0.908	0.911	0.919
wi	0.939	0.940	0.921	0.933	0.938	0.954
wr	0.480	0.494	0.468	0.490	0.480	0.494
ww	0.464	0.469	0.463	0.466	0.474	0.457
y	0.359	0.372	0.323	0.346	0.360	0.357
rank	3.29	3.00	3.43	3.71	2.46	3.21

Table A.7: Extended microaveraged F-score values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 20$ and $\gamma_\omega = 20$.

	Ψ_{MV}	Ψ_{RF}	$\Psi_{0.0}$	$\Psi_{0.3}$	$\Psi_{0.7}$	$\Psi_{1.0}$
aa	0.484	0.500	0.388	0.493	0.476	0.484
ap	0.856	0.824	0.628	0.907	0.846	0.877
ba	0.676	0.687	0.550	0.669	0.684	0.717
bi	0.723	0.721	0.721	0.726	0.700	0.741
bu	0.580	0.567	0.607	0.617	0.576	0.553
c	0.759	0.808	0.727	0.707	0.692	0.675
d	0.927	0.931	0.903	0.929	0.938	0.939
e	0.442	0.451	0.418	0.498	0.449	0.483
h	0.721	0.708	0.658	0.688	0.790	0.726
io	0.844	0.842	0.861	0.793	0.767	0.794
ir	0.950	0.948	0.941	0.973	0.972	0.960
me	0.644	0.668	0.598	0.543	0.660	0.683
po	0.916	0.915	0.687	0.925	0.891	0.929
ph	0.775	0.774	0.778	0.761	0.776	0.770
pi	0.736	0.735	0.707	0.761	0.713	0.731
ri	0.737	0.737	0.750	0.734	0.737	0.740
sb	0.707	0.712	0.707	0.715	0.707	0.716
se	0.936	0.935	0.928	0.932	0.946	0.933
tw	0.737	0.737	0.740	0.730	0.734	0.738
te	0.395	0.397	0.387	0.399	0.385	0.386
th	0.969	0.970	0.970	0.969	0.973	0.965
ti	0.781	0.774	0.774	0.783	0.765	0.785
wd	0.909	0.908	0.903	0.938	0.894	0.919
wi	0.939	0.940	0.953	0.931	0.935	0.947
wr	0.480	0.494	0.449	0.494	0.456	0.497
ww	0.464	0.469	0.472	0.449	0.453	0.481
y	0.359	0.372	0.301	0.363	0.345	0.383
rank	3.18	3.00	3.96	3.11	3.75	2.46

Table A.8: Extended microaveraged F-score values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 10$ and $\gamma_\omega = 10$.

	Ψ_{MV}	Ψ_{RF}	$\Psi_{0.0}$	$\Psi_{0.3}$	$\Psi_{0.7}$
aa	0.484	0.500	0.461	0.510	0.444
ap	0.856	0.824	0.752	0.832	0.849
ba	0.676	0.687	0.610	0.640	0.661
bi	0.723	0.721	0.724	0.710	0.729
bu	0.580	0.567	0.580	0.605	0.588
c	0.759	0.808	0.731	0.683	0.674
d	0.927	0.931	0.939	0.931	0.941
e	0.442	0.451	0.468	0.462	0.442
h	0.721	0.708	0.734	0.755	0.696
io	0.844	0.842	0.825	0.847	0.778
ir	0.950	0.948	0.939	0.964	0.946
me	0.644	0.668	0.703	0.683	0.818
po	0.916	0.915	0.741	0.922	0.924
ph	0.775	0.774	0.779	0.775	0.776
pi	0.736	0.735	0.756	0.731	0.750
ri	0.737	0.737	0.737	0.743	0.735
sb	0.707	0.712	0.718	0.699	0.708
se	0.936	0.935	0.932	0.923	0.937
tw	0.737	0.737	0.734	0.737	0.740
te	0.395	0.397	0.389	0.395	0.394
th	0.969	0.970	0.974	0.973	0.967
ti	0.781	0.774	0.788	0.764	0.763
wd	0.909	0.908	0.915	0.930	0.916
wi	0.939	0.940	0.948	0.937	0.939
wr	0.480	0.494	0.491	0.493	0.478
ww	0.464	0.469	0.443	0.464	0.466
y	0.359	0.372	0.355	0.350	0.381
rank	3.04	2.96	3.00	3.04	3.04

A.3. Macroaveraged F-score

Table A.9: Extended macroaveraged F-score values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 5$ and $\gamma_\omega = 5$.

	Ψ_{MV}	Ψ_{RF}	$\Psi_{0.0}$	$\Psi_{0.3}$	$\Psi_{0.7}$	$\Psi_{1.0}$
aa	0.204	0.196	0.201	0.173	0.172	0.230
ap	0.737	0.688	0.701	0.796	0.805	0.691
ba	0.471	0.477	0.466	0.485	0.498	0.450
bi	0.710	0.708	0.708	0.713	0.720	0.704
bu	0.566	0.555	0.563	0.522	0.585	0.547
c	0.768	0.817	0.848	0.735	0.889	0.830
d	0.927	0.931	0.932	0.937	0.938	0.916
e	0.132	0.171	0.107	0.180	0.110	0.147
h	0.588	0.573	0.513	0.586	0.539	0.631
io	0.833	0.829	0.833	0.849	0.860	0.832
ir	0.951	0.948	0.952	0.960	0.933	0.897
me	0.644	0.667	0.714	0.575	0.655	0.649
po	0.478	0.478	0.470	0.480	0.474	0.483
ph	0.726	0.726	0.722	0.742	0.717	0.717
pi	0.710	0.705	0.707	0.742	0.700	0.739
ri	0.748	0.746	0.743	0.745	0.740	0.738
sb	0.684	0.691	0.673	0.672	0.686	0.682
se	0.491	0.487	0.484	0.484	0.531	0.482
tw	0.737	0.737	0.745	0.740	0.734	0.729
te	0.390	0.390	0.395	0.388	0.388	0.385
th	0.830	0.829	0.825	0.824	0.834	0.829
ti	0.745	0.733	0.739	0.745	0.734	0.744
wd	0.902	0.901	0.902	0.876	0.890	0.894
wi	0.933	0.934	0.924	0.931	0.923	0.936
wr	0.221	0.244	0.227	0.241	0.217	0.204
ww	0.173	0.191	0.184	0.186	0.185	0.218
y	0.235	0.242	0.209	0.220	0.240	0.227
rank	2.89	3.00	3.46	3.07	3.11	3.61

Table A.10: Extended macroaveraged F-score values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 20$ and $\gamma_\omega = 5$.

	Ψ_{MV}	Ψ_{RF}	$\Psi_{0.0}$	$\Psi_{0.3}$	$\Psi_{0.7}$	$\Psi_{1.0}$
aa	0.204	0.196	0.262	0.211	0.223	0.185
ap	0.737	0.688	0.777	0.746	0.833	0.755
ba	0.471	0.477	0.442	0.464	0.478	0.481
bi	0.710	0.708	0.720	0.702	0.765	0.717
bu	0.566	0.555	0.605	0.537	0.617	0.484
c	0.768	0.817	0.726	0.669	0.774	0.832
d	0.927	0.931	0.929	0.927	0.919	0.919
e	0.132	0.171	0.107	0.130	0.209	0.092
h	0.588	0.573	0.613	0.536	0.648	0.535
io	0.833	0.829	0.875	0.795	0.858	0.780
ir	0.951	0.948	0.966	0.983	0.992	0.942
me	0.644	0.667	0.601	0.722	0.794	0.646
po	0.478	0.478	0.459	0.479	0.475	0.474
ph	0.726	0.726	0.720	0.737	0.710	0.732
pi	0.710	0.705	0.709	0.707	0.752	0.714
ri	0.748	0.746	0.745	0.747	0.753	0.749
sb	0.684	0.691	0.697	0.677	0.687	0.683
se	0.491	0.487	0.485	0.516	0.531	0.485
tw	0.737	0.737	0.733	0.744	0.740	0.729
te	0.390	0.390	0.399	0.391	0.386	0.407
th	0.830	0.829	0.808	0.840	0.842	0.841
ti	0.745	0.733	0.737	0.742	0.728	0.752
wd	0.902	0.901	0.864	0.896	0.903	0.913
wi	0.933	0.934	0.916	0.925	0.934	0.948
wr	0.221	0.244	0.224	0.241	0.227	0.225
ww	0.173	0.191	0.182	0.188	0.192	0.175
y	0.235	0.242	0.193	0.212	0.244	0.225
rank	3.50	3.39	3.75	3.54	2.07	3.46

Table A.11: Extended macroaveraged F-score values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 20$ and $\gamma_\omega = 20$.

	Ψ_{MV}	Ψ_{RF}	$\Psi_{0.0}$	$\Psi_{0.3}$	$\Psi_{0.7}$	$\Psi_{1.0}$
aa	0.204	0.196	0.195	0.201	0.246	0.195
ap	0.737	0.688	0.514	0.764	0.748	0.681
ba	0.471	0.477	0.481	0.463	0.475	0.493
bi	0.710	0.708	0.705	0.720	0.686	0.734
bu	0.566	0.555	0.616	0.609	0.590	0.543
c	0.768	0.817	0.748	0.724	0.667	0.669
d	0.927	0.931	0.904	0.928	0.938	0.937
e	0.132	0.171	0.143	0.165	0.188	0.114
h	0.588	0.573	0.597	0.542	0.638	0.617
io	0.833	0.829	0.847	0.768	0.775	0.784
ir	0.951	0.948	0.942	0.971	0.973	0.961
me	0.644	0.667	0.705	0.649	0.670	0.698
po	0.478	0.478	0.484	0.480	0.471	0.482
ph	0.726	0.726	0.729	0.710	0.728	0.715
pi	0.710	0.705	0.708	0.738	0.691	0.709
ri	0.748	0.746	0.761	0.746	0.750	0.749
sb	0.684	0.691	0.686	0.695	0.684	0.690
se	0.491	0.487	0.544	0.505	0.486	0.483
tw	0.737	0.737	0.740	0.731	0.734	0.738
te	0.390	0.390	0.392	0.412	0.380	0.389
th	0.830	0.829	0.836	0.821	0.864	0.817
ti	0.745	0.733	0.744	0.730	0.712	0.741
wd	0.902	0.901	0.895	0.933	0.888	0.913
wi	0.933	0.934	0.947	0.927	0.927	0.943
wr	0.221	0.244	0.235	0.285	0.230	0.264
ww	0.173	0.191	0.235	0.176	0.187	0.176
y	0.235	0.242	0.193	0.236	0.201	0.216
rank	3.50	3.36	2.86	3.25	3.57	3.32

Table A.12: Extended macroaveraged F-score values and Friedman rank of majority voting, random forest and integrated classifiers for $\gamma_B = 10$ and $\gamma_\omega = 10$.

	Ψ_{MV}	Ψ_{RF}	$\Psi_{0.0}$	$\Psi_{0.3}$	$\Psi_{0.7}$	$\Psi_{1.0}$
aa	0.204	0.196	0.232	0.178	0.180	0.236
ap	0.737	0.688	0.735	0.787	0.836	0.607
ba	0.471	0.477	0.496	0.459	0.459	0.461
bi	0.710	0.708	0.711	0.697	0.710	0.696
bu	0.566	0.555	0.547	0.593	0.571	0.569
c	0.768	0.817	0.767	0.688	0.683	0.752
d	0.927	0.931	0.939	0.931	0.941	0.933
e	0.132	0.171	0.148	0.139	0.104	0.201
h	0.588	0.573	0.629	0.642	0.542	0.613
io	0.833	0.829	0.818	0.825	0.759	0.779
ir	0.951	0.948	0.950	0.962	0.945	0.950
me	0.644	0.667	0.731	0.646	0.815	0.607
po	0.478	0.478	0.494	0.479	0.480	0.483
ph	0.726	0.726	0.731	0.725	0.728	0.722
pi	0.710	0.705	0.726	0.710	0.717	0.704
ri	0.748	0.746	0.747	0.755	0.746	0.741
sb	0.684	0.691	0.696	0.679	0.679	0.671
se	0.491	0.487	0.561	0.509	0.484	0.481
tw	0.737	0.737	0.734	0.737	0.740	0.740
te	0.390	0.390	0.396	0.377	0.385	0.388
th	0.830	0.829	0.836	0.830	0.814	0.828
ti	0.745	0.733	0.758	0.726	0.728	0.742
wd	0.902	0.901	0.912	0.927	0.907	0.892
wi	0.933	0.934	0.943	0.930	0.934	0.944
wr	0.221	0.244	0.280	0.290	0.212	0.237
ww	0.173	0.191	0.176	0.178	0.188	0.170
y	0.235	0.242	0.260	0.247	0.257	0.240
rank	3.25	3.21	2.11	3.11	3.50	3.79

Appendix B

Extended results for the second algorithm

This appendix contains extended results for the article presented in the section 4.2. It is divided regarding quality measures.

B.1. No displacements

B.1.1. Accuracy

Table B.1: Extended ACC values and Friedman rank of majority voting, random forest and integrated classifiers for proportional weighting function.

	Ψ_{MV}	Ψ_{RF}	Ψ^{vol}_{wMV}	Ψ^{vol}_{20}	Ψ^{vol}_{40}	Ψ^{vol}_{60}
ap	0.816	0.805	0.889	0.829	0.668	0.829
ba	0.900	0.909	0.945	0.951	0.720	0.951
bi	0.726	0.724	0.730	0.727	0.704	0.727
bu	0.558	0.520	0.541	0.628	0.606	0.618
c	0.830	0.821	0.864	0.909	0.773	0.909
d	0.934	0.924	0.935	0.952	0.892	0.952
e	0.604	0.604	0.580	0.580	0.580	0.580
h	0.678	0.701	0.694	0.696	0.649	0.696
io	0.840	0.864	0.818	0.818	0.834	0.818
me	0.643	0.591	0.691	0.655	0.577	0.655
po	0.929	0.929	0.907	0.907	0.719	0.907
ph	0.771	0.774	0.775	0.763	0.741	0.763
pi	0.753	0.740	0.778	0.739	0.626	0.739
ri	0.732	0.734	0.739	0.739	0.706	0.739
sb	0.717	0.716	0.705	0.395	0.705	0.395
se	0.929	0.929	0.929	0.929	0.904	0.929
tw	0.740	0.742	0.744	0.752	0.719	0.752
te	0.888	0.881	0.852	0.882	0.902	0.882
th	0.981	0.985	0.981	0.689	0.971	0.689
ti	0.784	0.777	0.801	0.808	0.790	0.808
wd	0.903	0.903	0.879	0.897	0.869	0.897
wi	0.946	0.955	0.954	0.948	0.948	0.948
wr	0.567	0.594	0.558	0.567	0.544	0.567
ww	0.630	0.627	0.637	0.642	0.626	0.642
y	0.683	0.648	0.611	0.630	0.509	0.630
rank	2.43	2.46	2.21	2.07	3.68	2.11

Table B.2: Extended ACC values and Friedman rank of majority voting, random forest and integrated classifiers for inversely proportional weighting function.

	Ψ_{MV}	Ψ_{RF}	Ψ^{inv}_{wMV}	Ψ^{inv}_{20}	Ψ^{inv}_{40}	Ψ^{inv}_{60}
ap	0.816	0.805	0.864	0.829	0.864	0.829
ba	0.900	0.909	0.951	0.951	0.951	0.951
bi	0.726	0.724	0.727	0.727	0.727	0.727
bu	0.558	0.520	0.562	0.605	0.542	0.605
c	0.830	0.821	0.864	0.909	0.818	0.909
d	0.934	0.924	0.949	0.946	0.949	0.946
e	0.604	0.604	0.580	0.580	0.580	0.580
h	0.678	0.701	0.683	0.696	0.683	0.696
io	0.840	0.864	0.818	0.818	0.818	0.818
me	0.643	0.591	0.726	0.691	0.726	0.691
po	0.929	0.929	0.907	0.907	0.907	0.907
ph	0.771	0.774	0.779	0.773	0.779	0.773
pi	0.753	0.740	0.744	0.729	0.724	0.727
ri	0.732	0.734	0.738	0.736	0.738	0.736
sb	0.717	0.716	0.394	0.404	0.394	0.404
se	0.929	0.929	0.929	0.930	0.929	0.930
tw	0.740	0.742	0.741	0.748	0.741	0.748
te	0.888	0.881	0.838	0.869	0.838	0.869
th	0.981	0.985	0.093	0.960	0.093	0.960
ti	0.784	0.777	0.808	0.808	0.808	0.808
wd	0.903	0.903	0.865	0.878	0.865	0.878
wi	0.946	0.955	0.948	0.948	0.948	0.948
wr	0.567	0.594	0.561	0.559	0.561	0.554
ww	0.630	0.627	0.641	0.642	0.641	0.642
y	0.683	0.648	0.681	0.666	0.681	0.666
rank	2.21	2.29	2.07	1.96	2.39	2.04

B.1.2. F-score

Table B.3: Extended F-score values and Friedman rank of majority voting, random forest and integrated classifiers for proportional weighting function.

	Ψ_{MV}	Ψ_{RF}	Ψ^{vol}_{wMV}	Ψ^{vol}_{20}	Ψ^{vol}_{40}	Ψ^{vol}_{60}
ap	0.884	0.880	0.935	0.896	0.773	0.896
ba	0.946	0.952	0.971	0.975	0.836	0.975
bi	0.771	0.770	0.774	0.768	0.744	0.768
bu	0.627	0.607	0.602	0.702	0.682	0.692
c	0.810	0.783	0.895	0.933	0.813	0.933
d	0.926	0.914	0.922	0.943	0.870	0.943
e	0.000	0.000	0.000	0.000	0.000	0.000
h	0.181	0.285	0.316	0.278	0.229	0.278
io	0.756	0.790	0.595	0.629	0.751	0.629
me	0.626	0.609	0.675	0.653	0.556	0.653
po	0.963	0.963	0.951	0.951	0.833	0.951
ph	0.609	0.613	0.622	0.582	0.604	0.582
pi	0.823	0.808	0.843	0.813	0.721	0.813
ri	0.767	0.770	0.759	0.773	0.741	0.773
sb	0.795	0.792	0.786	0.000	0.787	0.000
se	0.000	0.000	0.000	0.000	0.146	0.000
tw	0.746	0.741	0.739	0.755	0.719	0.755
te	0.917	0.915	0.911	0.911	0.922	0.911
th	0.811	0.853	0.822	0.190	0.740	0.190
ti	0.517	0.543	0.594	0.563	0.564	0.563
wd	0.920	0.920	0.904	0.919	0.889	0.919
wi	0.923	0.938	0.927	0.918	0.920	0.918
wr	0.502	0.533	0.459	0.515	0.516	0.515
ww	0.462	0.439	0.523	0.493	0.471	0.493
y	0.764	0.709	0.744	0.726	0.632	0.726
rank	2.50	2.54	2.29	2.46	3.32	2.50

Table B.4: Extended F-score values and Friedman rank of majority voting, random forest and integrated classifiers for inversely proportional weighting function.

	Ψ_{MV}	Ψ_{RF}	Ψ^{inv}_{wMV}	Ψ^{inv}_{20}	Ψ^{inv}_{40}	Ψ^{inv}_{60}
ap	0.884	0.880	0.919	0.896	0.919	0.896
ba	0.946	0.952	0.975	0.975	0.975	0.975
bi	0.771	0.770	0.768	0.768	0.768	0.768
bu	0.627	0.607	0.652	0.699	0.632	0.699
c	0.810	0.783	0.895	0.933	0.851	0.933
d	0.926	0.914	0.941	0.938	0.941	0.938
e	0.000	0.000	0.000	0.000	0.000	0.000
h	0.181	0.285	0.229	0.278	0.229	0.278
io	0.756	0.790	0.629	0.629	0.629	0.629
me	0.626	0.609	0.712	0.675	0.712	0.675
po	0.963	0.963	0.951	0.951	0.951	0.951
ph	0.609	0.613	0.634	0.613	0.634	0.613
pi	0.823	0.808	0.805	0.797	0.796	0.803
ri	0.767	0.770	0.762	0.770	0.762	0.770
sb	0.795	0.792	0.000	0.030	0.000	0.030
se	0.000	0.000	0.000	0.031	0.000	0.031
tw	0.746	0.741	0.740	0.749	0.740	0.749
te	0.917	0.915	0.884	0.905	0.884	0.905
th	0.811	0.853	0.109	0.554	0.109	0.554
ti	0.517	0.543	0.563	0.563	0.563	0.563
wd	0.920	0.920	0.894	0.903	0.894	0.903
wi	0.923	0.938	0.918	0.918	0.918	0.918
wr	0.502	0.533	0.516	0.494	0.559	0.503
ww	0.462	0.439	0.520	0.497	0.520	0.497
y	0.764	0.709	0.783	0.764	0.783	0.764
rank	2.21	2.18	2.18	2.07	2.29	1.96

B.1.3. Area under curve

Table B.5: Extended AUC values and Friedman rank of majority voting, random forest and integrated classifiers for proportional weighting function.

	Ψ_{MV}	Ψ_{RF}	Ψ^{vol}_{wMV}	Ψ^{vol}_{20}	Ψ^{vol}_{40}	Ψ^{vol}_{60}
ap	0.722	0.671	0.729	0.691	0.590	0.691
ba	0.492	0.500	0.500	0.500	0.381	0.500
bi	0.726	0.725	0.726	0.728	0.710	0.728
bu	0.547	0.512	0.533	0.603	0.584	0.594
c	0.848	0.865	0.875	0.906	0.804	0.906
d	0.936	0.927	0.933	0.955	0.889	0.955
e	0.500	0.500	0.500	0.500	0.500	0.500
h	0.513	0.553	0.575	0.569	0.536	0.569
io	0.809	0.841	0.743	0.751	0.819	0.751
me	0.654	0.597	0.733	0.671	0.567	0.671
po	0.500	0.500	0.500	0.500	0.492	0.500
ph	0.720	0.722	0.727	0.701	0.714	0.701
pi	0.709	0.706	0.731	0.693	0.587	0.693
ri	0.732	0.734	0.739	0.740	0.708	0.740
sb	0.668	0.669	0.654	0.499	0.654	0.499
se	0.499	0.499	0.500	0.500	0.539	0.500
tw	0.740	0.742	0.745	0.752	0.720	0.752
te	0.862	0.813	0.706	0.842	0.888	0.842
th	0.900	0.942	0.915	0.648	0.845	0.648
ti	0.674	0.685	0.712	0.695	0.694	0.695
wd	0.902	0.898	0.876	0.890	0.878	0.890
wi	0.941	0.949	0.945	0.937	0.940	0.937
wr	0.569	0.597	0.559	0.569	0.543	0.569
ww	0.592	0.585	0.614	0.607	0.591	0.607
y	0.643	0.641	0.528	0.581	0.459	0.581
rank	2.5	2.46	2.14	2.25	3.36	2.29

Table B.6: Extended AUC values and Friedman rank of majority voting, random forest and integrated classifiers for inversely proportional weighting function.

	Ψ_{MV}	Ψ_{RF}	Ψ^{inv}_{wMV}	Ψ^{inv}_{20}	Ψ^{inv}_{40}	Ψ^{inv}_{60}
ap	0.722	0.671	0.714	0.691	0.714	0.691
ba	0.492	0.500	0.500	0.500	0.500	0.500
bi	0.726	0.725	0.728	0.728	0.728	0.728
bu	0.547	0.512	0.529	0.563	0.513	0.563
c	0.848	0.865	0.875	0.906	0.844	0.906
d	0.936	0.927	0.953	0.949	0.953	0.949
e	0.500	0.500	0.500	0.500	0.500	0.500
h	0.513	0.553	0.551	0.569	0.551	0.569
io	0.809	0.841	0.751	0.751	0.751	0.751
me	0.654	0.597	0.758	0.733	0.758	0.733
po	0.500	0.500	0.500	0.500	0.500	0.500
ph	0.720	0.722	0.736	0.721	0.736	0.721
pi	0.709	0.706	0.735	0.707	0.696	0.685
ri	0.732	0.734	0.740	0.738	0.740	0.738
sb	0.668	0.669	0.498	0.506	0.498	0.506
se	0.499	0.499	0.500	0.508	0.500	0.508
tw	0.740	0.742	0.742	0.748	0.742	0.748
te	0.862	0.813	0.816	0.834	0.816	0.834
th	0.900	0.942	0.502	0.710	0.502	0.710
ti	0.674	0.685	0.695	0.695	0.695	0.695
wd	0.902	0.898	0.858	0.879	0.858	0.879
wi	0.941	0.949	0.937	0.937	0.937	0.937
wr	0.569	0.597	0.564	0.560	0.557	0.553
ww	0.592	0.585	0.615	0.608	0.615	0.608
y	0.643	0.641	0.579	0.587	0.579	0.587
rank	2.18	2.29	2.00	1.93	2.36	2.11

B.2. 5 displacements

B.2.1. Accuracy

Table B.7: Extended ACC values and Friedman rank of majority voting, random forest and integrated classifiers for proportional weighting function.

	Ψ_{MV}	Ψ_{RF}	Ψ^{vol}_{wMV}	Ψ^{vol}_{20}	Ψ^{vol}_{40}	Ψ^{vol}_{60}
ap	0.816	0.805	0.835	0.810	0.692	0.810
ba	0.900	0.909	0.885	0.848	0.696	0.848
bi	0.726	0.724	0.718	0.725	0.694	0.725
bu	0.558	0.520	0.472	0.517	0.563	0.524
c	0.830	0.821	0.867	0.750	0.700	0.750
d	0.934	0.924	0.910	0.926	0.908	0.926
e	0.604	0.604	0.628	0.628	0.628	0.628
h	0.678	0.701	0.717	0.671	0.566	0.671
io	0.840	0.864	0.795	0.745	0.808	0.745
me	0.643	0.591	0.569	0.611	0.669	0.611
po	0.929	0.929	0.952	0.952	0.739	0.952
ph	0.771	0.774	0.765	0.764	0.748	0.764
pi	0.753	0.740	0.770	0.766	0.663	0.766
ri	0.732	0.734	0.736	0.730	0.720	0.730
sb	0.717	0.716	0.686	0.520	0.704	0.520
se	0.929	0.929	0.932	0.930	0.902	0.930
tw	0.740	0.742	0.729	0.733	0.714	0.733
te	0.888	0.881	0.870	0.874	0.923	0.874
th	0.981	0.985	0.969	0.818	0.977	0.818
ti	0.784	0.777	0.771	0.759	0.759	0.759
wd	0.903	0.903	0.909	0.916	0.796	0.916
wi	0.946	0.955	0.935	0.945	0.945	0.945
wr	0.567	0.594	0.585	0.563	0.556	0.561
ww	0.630	0.627	0.622	0.609	0.543	0.609
y	0.683	0.648	0.735	0.720	0.571	0.720
rank	1.96	2.18	2.54	3.00	3.54	2.96

Table B.8: Extended ACC values and Friedman rank of majority voting, random forest and integrated classifiers for inversely proportional weighting function.

	Ψ_{MV}	Ψ_{RF}	Ψ^{inv}_{wMV}	Ψ^{inv}_{20}	Ψ^{inv}_{40}	Ψ^{inv}_{60}
ap	0.816	0.805	0.816	0.804	0.816	0.804
ba	0.900	0.909	0.848	0.848	0.848	0.848
bi	0.726	0.724	0.725	0.725	0.725	0.725
bu	0.558	0.520	0.518	0.531	0.517	0.531
c	0.830	0.821	0.773	0.750	0.773	0.750
d	0.934	0.924	0.924	0.925	0.924	0.925
e	0.604	0.604	0.628	0.628	0.628	0.628
h	0.678	0.701	0.673	0.668	0.673	0.668
io	0.840	0.864	0.755	0.752	0.755	0.752
me	0.643	0.591	0.594	0.611	0.594	0.611
po	0.929	0.929	0.952	0.952	0.952	0.952
ph	0.771	0.774	0.760	0.767	0.760	0.767
pi	0.753	0.740	0.769	0.771	0.762	0.772
ri	0.732	0.734	0.729	0.731	0.729	0.731
sb	0.717	0.716	0.418	0.615	0.418	0.615
se	0.929	0.929	0.930	0.930	0.930	0.930
tw	0.740	0.742	0.729	0.733	0.729	0.733
te	0.888	0.881	0.844	0.884	0.844	0.884
th	0.981	0.985	0.475	0.933	0.475	0.933
ti	0.784	0.777	0.759	0.759	0.759	0.759
wd	0.903	0.903	0.907	0.919	0.907	0.919
wi	0.946	0.955	0.945	0.945	0.945	0.945
wr	0.567	0.594	0.572	0.564	0.567	0.561
ww	0.630	0.627	0.612	0.612	0.612	0.612
y	0.683	0.648	0.703	0.712	0.703	0.712
rank	1.75	2.00	2.64	2.36	2.75	2.36

B.2.2. F-score

Table B.9: Extended F-score values and Friedman rank of majority voting, random forest and integrated classifiers for proportional weighting function.

	Ψ_{MV}	Ψ_{RF}	Ψ^{vol}_{wMV}	Ψ^{vol}_{20}	Ψ^{vol}_{40}	Ψ^{vol}_{60}
ap	0.884	0.880	0.898	0.876	0.768	0.876
ba	0.946	0.952	0.939	0.917	0.820	0.917
bi	0.771	0.770	0.773	0.774	0.729	0.774
bu	0.627	0.607	0.580	0.579	0.538	0.579
c	0.810	0.783	0.889	0.690	0.625	0.690
d	0.926	0.914	0.903	0.920	0.898	0.920
e	0.000	0.000	0.000	0.000	0.000	0.000
h	0.181	0.285	0.235	0.143	0.251	0.143
io	0.756	0.790	0.643	0.542	0.736	0.542
me	0.626	0.609	0.557	0.577	0.650	0.577
po	0.963	0.963	0.975	0.975	0.848	0.975
ph	0.609	0.613	0.618	0.604	0.559	0.604
pi	0.823	0.808	0.835	0.835	0.753	0.835
ri	0.767	0.770	0.765	0.766	0.762	0.766
sb	0.795	0.792	0.777	0.314	0.786	0.314
se	0.000	0.000	0.000	0.000	0.075	0.000
tw	0.746	0.741	0.728	0.741	0.714	0.741
te	0.917	0.915	0.901	0.902	0.938	0.902
th	0.811	0.853	0.657	0.451	0.771	0.451
ti	0.517	0.543	0.546	0.472	0.472	0.472
wd	0.920	0.920	0.920	0.926	0.818	0.926
wi	0.923	0.938	0.917	0.927	0.923	0.927
wr	0.502	0.533	0.502	0.500	0.542	0.496
ww	0.462	0.439	0.474	0.423	0.365	0.423
y	0.764	0.709	0.812	0.776	0.600	0.776
rank	2.00	2.04	2.39	2.89	3.25	2.96

Table B.10: Extended F-score values and Friedman rank of majority voting, random forest and integrated classifiers for inversely proportional weighting function.

	Ψ_{MV}	Ψ_{RF}	Ψ^{inv}_{wMV}	Ψ^{inv}_{20}	Ψ^{inv}_{40}	Ψ^{inv}_{60}
ap	0.884	0.880	0.881	0.871	0.881	0.871
ba	0.946	0.952	0.917	0.917	0.917	0.917
bi	0.771	0.770	0.774	0.774	0.774	0.774
bu	0.627	0.607	0.586	0.586	0.584	0.586
c	0.810	0.783	0.724	0.686	0.724	0.686
d	0.926	0.914	0.918	0.919	0.918	0.919
e	0.000	0.000	0.000	0.000	0.000	0.000
h	0.181	0.285	0.143	0.133	0.143	0.133
io	0.756	0.790	0.534	0.547	0.534	0.547
me	0.626	0.609	0.570	0.573	0.570	0.573
po	0.963	0.963	0.975	0.975	0.975	0.975
ph	0.609	0.613	0.598	0.610	0.598	0.610
pi	0.823	0.808	0.836	0.838	0.832	0.838
ri	0.767	0.770	0.764	0.767	0.764	0.767
sb	0.795	0.792	0.045	0.563	0.045	0.563
se	0.000	0.000	0.007	0.000	0.007	0.000
tw	0.746	0.741	0.738	0.742	0.738	0.742
te	0.917	0.915	0.877	0.910	0.877	0.910
th	0.811	0.853	0.277	0.550	0.277	0.550
ti	0.517	0.543	0.472	0.472	0.472	0.472
wd	0.920	0.920	0.919	0.929	0.919	0.929
wi	0.923	0.938	0.927	0.927	0.927	0.927
wr	0.502	0.533	0.510	0.498	0.505	0.491
ww	0.462	0.439	0.422	0.425	0.422	0.425
y	0.764	0.709	0.758	0.768	0.758	0.768
rank	1.75	1.86	2.68	2.29	2.79	2.32

B.2.3. Area under curve

Table B.11: Extended AUC values and Friedman rank of majority voting, random forest and integrated classifiers for proportional weighting function.

	Ψ_{MV}	Ψ_{RF}	Ψ^{vol}_{wMV}	Ψ^{vol}_{20}	Ψ^{vol}_{40}	Ψ^{vol}_{60}
ap	0.722	0.671	0.712	0.757	0.745	0.757
ba	0.492	0.500	0.500	0.484	0.395	0.484
bi	0.726	0.725	0.711	0.725	0.724	0.725
bu	0.547	0.512	0.483	0.531	0.568	0.537
c	0.848	0.865	0.875	0.790	0.771	0.790
d	0.936	0.927	0.909	0.928	0.909	0.928
e	0.500	0.500	0.500	0.500	0.500	0.500
h	0.513	0.553	0.537	0.476	0.468	0.476
io	0.809	0.841	0.737	0.679	0.787	0.679
me	0.654	0.597	0.564	0.616	0.699	0.616
po	0.500	0.500	0.500	0.500	0.447	0.500
ph	0.720	0.722	0.729	0.719	0.686	0.719
pi	0.709	0.706	0.728	0.714	0.618	0.714
ri	0.732	0.734	0.734	0.727	0.717	0.727
sb	0.668	0.669	0.632	0.563	0.655	0.563
se	0.499	0.499	0.500	0.499	0.511	0.499
tw	0.740	0.742	0.728	0.733	0.714	0.733
te	0.862	0.813	0.850	0.850	0.911	0.850
th	0.900	0.942	0.813	0.744	0.919	0.744
ti	0.674	0.685	0.683	0.652	0.652	0.652
wd	0.902	0.898	0.910	0.917	0.794	0.917
wi	0.941	0.949	0.943	0.946	0.940	0.946
wr	0.569	0.597	0.592	0.564	0.558	0.562
ww	0.592	0.585	0.590	0.570	0.511	0.570
y	0.643	0.641	0.678	0.705	0.580	0.705
rank	2.18	2.14	2.46	2.79	3.21	2.79

Table B.12: Extended AUC values and Friedman rank of majority voting, random forest and integrated classifiers for inversely proportional weighting function.

	Ψ_{MV}	Ψ_{RF}	Ψ^{inv}_{wMV}	Ψ^{inv}_{20}	Ψ^{inv}_{40}	Ψ^{inv}_{60}
ap	0.722	0.671	0.761	0.753	0.761	0.753
ba	0.492	0.500	0.484	0.484	0.484	0.484
bi	0.726	0.725	0.725	0.725	0.725	0.725
bu	0.547	0.512	0.528	0.543	0.528	0.543
c	0.848	0.865	0.806	0.789	0.806	0.789
d	0.936	0.927	0.926	0.928	0.926	0.928
e	0.500	0.500	0.500	0.500	0.500	0.500
h	0.513	0.553	0.477	0.471	0.477	0.471
io	0.809	0.841	0.682	0.685	0.682	0.685
me	0.654	0.597	0.603	0.616	0.603	0.616
po	0.500	0.500	0.500	0.500	0.500	0.500
ph	0.720	0.722	0.714	0.722	0.714	0.722
pi	0.709	0.706	0.720	0.720	0.710	0.721
ri	0.732	0.734	0.726	0.728	0.726	0.728
sb	0.668	0.669	0.505	0.618	0.505	0.618
se	0.499	0.499	0.501	0.499	0.501	0.499
tw	0.740	0.742	0.729	0.733	0.729	0.733
te	0.862	0.813	0.811	0.861	0.811	0.861
th	0.900	0.942	0.608	0.815	0.608	0.815
ti	0.674	0.685	0.652	0.652	0.652	0.652
wd	0.902	0.898	0.907	0.918	0.907	0.918
wi	0.941	0.949	0.946	0.946	0.946	0.946
wr	0.569	0.597	0.574	0.566	0.568	0.562
ww	0.592	0.585	0.572	0.572	0.572	0.572
y	0.643	0.641	0.688	0.698	0.688	0.698
rank	1.86	1.93	2.57	2.29	2.68	2.29

Abstract

Popularity of artificial intelligence and machine learning methods is still growing and can be found in fields like cybersecurity, optimization, finance, medicine and healthcare, law, media or education.

Supervised learning is one of the methods of machine learning. The classification algorithms build a model, also called a classifier, with the use of labelled data building up the training set. The created model is then utilized to label the new, not yet labelled objects.

An important role among classification algorithms play ensemble classifiers. It was noticed that utilizing multiple models forming a system leads to improvement in classification performance.

Research problem

One such approach to classifier integration is by using their geometric representation expressed by the decision boundaries. Utilizing decision boundaries, which are the result of classifier training, allows for defining novel algorithms for model integration. These algorithms do not use the labels' probability vector or the labels themself.

In the dissertation, four novel algorithms using decision boundaries produced by decision trees are proposed.

The main goal of the study was to broaden the knowledge of diverse multiclassifier system creation methods based on their geometric features, proposal of new algorithms and their implementation as well as evaluation against referential techniques: majority voting and random forest. The research hypothesis of this dissertation is formulated as follows.

Research hypothesis. *Utilization of trained decision trees' decision boundaries allows for building an ensemble of classifiers with a greater value of performance quality measure than the multiclassifier system like random forest or majority voting using the same set of trained decision trees.*

Designed algorithms of classifier ensembling were compared to other ensemble techniques: (weighted) majority voting and random forest. The following classification performance measures were used to prove the research hypothesis: accuracy (ACC), Matthews Correlation Coefficient (MCC) and F-score for binary classification datasets and accuracy, precision, recall and F-score (three latter both micro– and macroaveraged) for the other problems. The experimental studies were performed using testing, open–source datasets from UCI and KEEL platforms. The proposed research hypothesis was verified by non–parametric statistical tests involving diverse classification quality measures.

Four algorithms for decision tree integration employing their geometric representation were developed, implemented and evaluated. Statistical analysis of the results obtained for the datasets mentioned indicates that the proposed methods outperform in many cases the referential integration techniques. In the dissertation it was proven, that the proposed algorithm is a generalization of weighted majority voting.

Achieved results

The first integration algorithm utilizes two distances in the feature space used as the coordinate system with the classification boundaries of trained classifiers projected. The first distance is calculated from the centroid and the second – from the decision boundary. The object under test is classified by the model of the trained decision tree itself. The centroid of the corresponding class is taken to compute the distance. Both distances were mapped with the Gaussian function. Several parameter values of the mapping function were studied. The additional mapping serves two purposes:

- The distances are normalized to the unit range: $[0, 1]$.
- The calculated value reflects the contribution of the distance from the decision boundary to the weight value.

The final decision was computed as the class label for which the linear combination of both distances is maximal. Several possible distribution parameters were examined in order to find the highest quality measure. The statistical tests indicated the significant

improvement in the classification quality measures: ACC and MCC of the presented technique in comparison with the referential ensemble methods.

The second proposal is an algorithm where the ensemble model is produced using the static space division. The feature space is split into equal subspaces, with the same number of divisions along every axis, whose label is determined by the classification of the middle point. For the particular label, the weight is computed for the subspaces based on the volume of classification regions. Classification regions are the cuboids with the highest volume spanning over the points of a single decision tree classification. Additional conference article presents a theoretical proof that the presented algorithm is the generalization of the weighted majority voting. The implementation was examined using the datasets. Statistically significant classification quality improvement was found, especially in the quality measures like ACC and MCC.

In the third method proposed, dynamic space partitioning was employed. The geometric representation of the trained model determines the division of the feature space. For every label, the weight is calculated based on the classification of the region itself and its neighbors. Half of the weight is assigned from the subspace itself and the weights of its neighbors sum up to the other half. The weights depend on the distances between the middle points of the neighboring subspaces. The statistical analysis was conducted and the performance improvement expressed with microaveraged precision, recall and F-score in comparison with the referential majority voting and random forest was found.

In the last work, similarly as in the previous one, dynamic division of feature space is applied and the subspaces' classification is influenced by the surrounding subspaces. However, additional improvements were introduced:

- Mean of the training points is used instead of the geometric middle point of the subspace in the definition of the distance between the regions. The distance between the averages is taken as the distance between the subspaces.
- Only the subspaces containing the objects among which the ones with the minor label made up at least 5% of all the objects in that subspace were taken during weight calculation.
- A wider range of neighbor subspaces is taken in case the filtering rejects the whole neighbor ring. The procedure is conducted until a non–empty ring is found.

The discussed algorithm was tested using the datasets mentioned against both referential methods (majority voting and random forest) and the previously presented, original algorithm. Statistically significant improvement is observed in the quality performance against all the reference methods. In particular for the following measures: accuracy,

micro– and macro–averaged F-score statistically significant difference was found, which proves the research hypothesis.

A consistent experimental setup is followed in all the studies. Comprehensive evaluation on multiple benchmarking datasets was conducted. Several classification quality measures are computed to account for diverse datasets' features, like imbalance ratio: ACC, MCC and F-score for binary and micro– and macro–averaged precision, recall and F-score for other problems. Spark with Scala was harnessed to implement every presented algorithm. The source code is hosted and publicly available on github for other researchers working on machine learning problems to verify: https://github.com/TAndronicus/dtree-merge, https://github.com/TAndronicus/dtree-merge-scoring, https://github.com/TAndronicus/dynamic-dtree, https://github.com/TAndronicus/dynamic-ring.

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

Streszczenie

Popularność wykorzystania metod sztucznej inteligencji oraz uczenia maszynowego jest stale rosnąca i zauważalna w takich dziedzinach jak cyberbezpieczeństwo, optymalizacja, finanse, medycyna, opieka zdrowotna, prawo, media czy też edukacja.

Klasyfikacja nadzorowana jest jedną z metod uczenia maszynowego. Algorytmy klasyfikacji nadzorowanej budują pewien model, który nazywany jest też klasyfikatorem, z wykorzystaniem poetykietowanych danych tworzących zbiór uczący. Utworzony model jest następnie wykorzystywany do nadawania etykiet nowym, niepoetykietowanym wcześniej obiektom.

Ważną rolę wśród algorytmów klasyfikacji nadzorowanej odgrywają klasyfikatory łączone. Zauważono, że użycie wielu modeli, które tworzą pewien zespoł prowadzi do poprawienia jakości klasyfikacji.

Problem badawczy

Jedno z podejść do łączenia klasyfikatorów zakłada użycie ich reprezentacji geometrycznej wyrażonej jako granice decyzyjne. Wykorzystanie granic decyzyjnych, które są efektem uczenia klasyfikatora pozwala na zdefiniowanie nowych algorytmów łączenia klasyfikatorów. Algorytmy te nie wykorzytują wektora prawdopodobieństw etykiet klas czy też samych etykiet klas.

W rozprawie zaproponowano 4 autorskie algorytmy wykorzystujące granice decyzyjne wyznaczone przez drzewa decyzyjne.

Głównym celem badań było poszerzenie wiedzy na temat różnorodnych systemów wielu klasyfikatorów opartych na ich geometrycznych właściwościach, zaprojekowanie i implementacja nowych algorytmów oraz ich ewaluacja w odniesieniu do metod referencyjnych: głosowania większościowego i lasu losowego. Hipoteza badawcza przyjęta w pracy sformułowana jest w następujący sposób.

Hipoteza badawcza. *Wykorzystanie granic decyzyjnych wytrenowanych drzew decyzyjnych pozwala na zbudowanie klasyfikatora łączonego o większej wartości miary jakości klasyfikacji niż klasyfikator łączony jakim jest las losowy lub głosowanie większościowe korzystające z tego samego zbioru wytrenowanych drzew decyzyjnych.*

Opracowane algorytmy łączenia klasyfikatorów zostały porównane z innymi algorytmami dedykowanymi do zespołu klasyfikatorów: (ważonego) głosowania większościowego oraz lasu losowego. Następujące miary jakości klasyfikacji zostały wykorzystane do udowodnienia hipotezy badawczej: dokładność (ACC), współczynnik korelacji Matthews'a (MCC) oraz współczynnik F1 dla zbiorów binarnych oraz dokładność, precyzja, czułość i wskaźnik F1 (trzy ostatnie w formie mikro– i makro–uśrednionej) dla pozostałych problemów. Przeprowadzono badania eksperymentalne z wykorzystaniem testowych baz danych na licencji open–source pochodzących z platform UCI oraz KEEL. Zaproponowana hipoteza badawcza została zweryfikowna z wykorzystaniem nieparametrycznych testów statystycznych, które uwzględniały różne miary jakości klasyfikacji.

Zaimplementowano i przetestowano cztery algorytmy integracji drzew decyzyjnych wykorzystujące ich reprezentację geometryczną. Analiza statystyczna otrzymanych wyników wskazuje, że zaproponowane metody w wielu przypadkach przewyższają w działaniu referencyjne techniki integracji. W rozprawie wykazano, że zaproponowany algorytm jest uogólnieniem ważonego głosowania większościowego.

Osiągnięte wyniki

Pierwszy algorytm integracji wykorzystuje dwie odległości w przestrzeni cech rozpatrywanej jako układ współrzędnych z wyróżnionymi granicami decyzyjnymi wytrenowanych klasyfikatorów. Pierwsza odległość liczona jest od centroidu, druga – od granicy decyzyjnej. Testowany obiekt klasyfikowany jest przez wytrenowany model drzewa decyzyjnego. Następnie centroid dla wybranej klasy determinuje wartość pierwszej odległości. Obie wartości mapowane są z użyciem funkcji Gaussa. Zbadano kilka zestawów parametrów funkcji mapującej, która pełni dwa zasadnicze zadania:

- Normalizacja obu odległości do wartości z zakresu $[0, 1]$.
- Obliczona wartość odzwierciedla wpływ odległości od granicy decyzyjnej na wagę.

Ostateczną odpowiedzią klasyfikatora jest etykieta klasy, której kombinacja liniowa obu odległości jest maksymalna. Kilka możliwych parametrów rozkładu funkcji mapującej

zostało przetestowanych w celu znalezienia najwyższej wartości miary jakości klasyfikacji. Testy statystyczne wskazują na znaczący wzrost wartości miar klasyfikacji ACC oraz MCC zaprezentowanej techniki w stosunku do referencyjnych metod łączenia.

Druga propozycja algorytmu korzysta ze statycznego podziału przestrzeni cech w procesie integracji. Przestrzeń cech podzielona jest na równe podprzestrzenie, z taką samą liczbą podziałów wzdłuż każdej z osi, których etykiety determinowane są przez klasyfikację ich punktów środkowych. Dla każdej podprzestrzeni dla danej etykiety obliczana jest waga zależna od objętości regionu klasyfikacji. Region klasyfikacji jest hiperkostką o maksymalnej objętości obejmującą punkty oznaczone tą samą etykietą przez dane drzewo decyzyjne. Dodatkowo podano teoretyczny dowód na to, że zaproponowana technika jest uogólnieniem ważonego głosowania większościowego. Implementacja algorytmu została sprawdzona z wykorzystaniem wspomnianych zbiorów danych. Zaobserwowano statystycznie znaczącą poprawę jakości klasyfikacji m. in. w takich miarach jak ACC i MCC.

W trzeciej zaproponowanej metodzie wykorzystano dynamiczny podział przestrzeni cech. Geometryczna reprezentacja wytrenowanych modeli determinuje podział przestrzeni cech. Dla każdej etykiety obliczana jest waga na podstawie klasyfikacji samej podprzestrzeni jak również jej sąsiadów. Połowę wagi stanowi wkład pochodzący z samej podprzestrzeni, wagi sąsiadujących podprzestrzeni sumują się natomiast do $\frac{1}{2}$. Wartości wag zależą od odległości między środkami sąsiadujących podprzestrzeni. Przeprowadzono analizę statystyczną i zaobserwowano poprawę w jakości klasyfikacji wyrażoną mikrouśrednioną precyzją, czułością i współczynnikiem F1 w porównaniu z referencyjnymi metodami: głosowaniem większościowym oraz lasem losowym.

W ostatniej pracy, podobnie jak w poprzedniej, wykorzystany jest dynamiczny podział przestrzeni cech, a wynik klasyfikacji w danej podprzestrzeni zależny jest od podprzestrzeni ją otaczających. Wprowadzono dodatkowe udoskonalenia:

- W definicji odległości między podprzestrzeniami wykorzystano średnią z punktów treningowych w danej podprzestrzeni zamiast jej geometrycznego środka. Odległość między średnimi jest jednocześnie odległością między podprzestrzeniami.
- W obliczaniu wagi rozpatrywane są jedynie podprzestrzenie, w których obiektów treningowych każdej klasy jest przynajmniej 5% wszystkich obiektów w danej podprzestrzeni.
- Większy zakres sąsiadujących podprzestrzeni jest analizowany w przypadku odfiltrowania całego pierścienia sąsiadów. Ta procedura powtarzana jest do momentu aż znaleziony zostanie niepusty pierścień.

Omawiany algorytm został w badaniach eksperymentalnych porównany z wykorzystaniem wspomnianych baz danych z referencyjnymi metodami (głosowaniem większościowym i lasem losowym) jak również zaprezentowanym wcześniej, pierwotnym algorytmem. Zaobserwowano statystycznie znaczącą poprawę w jakości klasyfikacji w stosunku do metod referencyjnych. W szczególności dla miar jakości klasyfikatorów: dokładności, mikro– i makrouśrednionego współczynnika F1 wykazana została statystycznie znacząca różnica, co dowodzi hipotezy badawczej.

We wszystkich badaniach zachowano spójną strukturę badań. Przeprowadzono wyczerpującą ewaluację z użyciem wielu testowych baz danych. Kilka miar jakości klasyfikacji zostało obliczonych mając na względzie różnorodną charakterystykę baz danych, na przykład współczynnik niezbalansowania: dokładność, współczynnik korelacji Matthews'a oraz współczynnik F1 dla binarnych oraz dokładność jak również mikro– i makrouśrednioną precyzję, czułość i współczynnik F1 dla pozostałych problemów. Do implementacji zaprezentowanych algorytmów wykorzystano framework Spark i język Scala. Kod źródłowy jest publicznie dostępny na portalu github, między innymi w celu weryfikacji opracowanych algorytmów przez innych badaczy zajmujących się problemami uczenia maszynowego: https://github.com/TAndronicus/dtree-merge, https://github.com/TAndronicus/dtree-merge-scoring, https://github.com/TAndronicus/dynamic-dtree, https://github.com/TAndronicus/dynamic-ring.

Ta praca została częściowo wsparta przez polskie Narodowe Centrum Nauki, nr grantu 2017/25/B/ST6/01750.