WROCŁAW UNIVERSITY OF SCIENCE AND TECHNOLOGY

Multi-objective optimization to train classifiers on feature subspaces

by

Joanna Klikowska

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Information and Communication Technology
Department of Systems and Computer Networks

May 2024

Acknowledgements

I am deeply indebted to my supervisor, prof. dr hab. Michał Woźniak for his invaluable patience and feedback.

Words cannot express my gratitude to my husband, dr inż. Jakub Klikowski supported me both scientifically and privately. He paved the way for my scientific and educational path and supported me at every stage of my journey.

I am also grateful to my coworkers from the Department of Computer Systems and Networks for their valuable comments on my diploma thesis.

Podziękowania

Przede wszystkim chciałabym podziękować mojemu promotorowi, prof. dr hab. Michałowi Woźniakowi za nieocenioną cierpliwość i wszelką pomoc w trakcie powstawania tej pracy.

Ogromne podziękowania należą się mojemu mężowi, dr inż. Jakubowi Klikowskiemu, który wspierał mnie w zakresie naukowym, jak i prywatnym. Przecierał szlak ścieżki naukowo-dydaktycznej i był wsparciem na każdym etapie mojej drogi.

Dziękuję także współpracownikom z Katedry Systemów i Sieci Komputerowych za cenne uwagi dotyczące mojej pracy dyplomowej.

Abbreviations

$5x2$ CV	Repeated Stratified K-Fold **5-splits** $\times$ **2-fold cross-validation**
AI	artificial intelligence
Attr.	number of attributes
AUC	area under curve
BAC	balanced accuracy
CR	crossover constant, parameter in DE
DE	differential evolution algorithm
$DE - Forest$	differential evolution forest
DT	CART decision tree classifier
$Ex.$	number of examples
FN	false negative - number of wrong classified positive examples
FNR	false negative rate
FP	false positive - number of wrong classified negative examples
FPR	false positive rate
FS	feature selection
$FSIRSVM$	feature selection imbalance ratio SVM
GA	genetic algorithm
$GA - a$	GA with objective function accuracy
$GA - ac$	GA with objective function accuracy and features' cost

$Gmean$	geometric mean of precision and recall
$Gmean_s$	geometric mean of specificity and recall
GNB	gaussian naive bayes classifier
ID	dataset identifier
IR	imbalance ratio
kNN	k-nearest neighbors classifier
LHS	latin hypercube sampling
$MOEA/D$	multi-objective evolutionary algorithm based on decomposition
$MOLO$	multi-objective local optimization forest
$MOLO-HG$	MOLO with optimization criteria *Hellinger* and *Gmean*
$MOLO-MD$	MOLO with optimization criteria *Margin* and *Diversity*
$MOLO-MDH$	MOLO with optimization criteria *Margin*, *Diversity* and *Hellinger*
$MOLO-HG\_h$	MOLO with optimization criteria *Hellinger* and *Gmean* using holdout
$MOLO-MD\_h$	MOLO with optimization criteria *Margin* and *Diversity*, using holdout
$MOLO-MDH\_h$	MOLO with optimization criteria *Margin*, *Diversity* and *Hellinger*, using holdout
MOO	multi-objective optimization
$MOOforest$	multi-objective optimization forest
$NSGA-II$	non-dominated sorting genetic algorithm II
PF	pareto front
PS	non-dominated solution set
$RandomFS$	random forest
$RandomFS_b$	random forest with bootstrapping
RBF	radial basis function kernel of SVM
RF	random forest
RS	random subspace ensemble
$SEMOOS$	SVM ensemble with multi-objective optimization selection

$SEMOOSb$	$SEMOOS$ with bootstrapping
$SEMOOSbp$	$SEMOOS$ with bootstrapping and pruning
SVM	support vector machine classifier
TN	true negative - number of correct classified negative examples
TNR	true negative rate (specificity)
TP	true positive - number of correct classified positive examples
TPR	true positive rate (recall, sensitivity)

Symbols

A_i	the i-th non-dominated set
B	number of bootstrapped subsets
b_i	ith subset from bootstrapping
C	regularization parameter of SVM
D	sum of distances
D_i	ith distribution
d	number of attributes (features)
d_H	Hellinger distance
dim	dimension of the searched space
dis	distance between root r and example
e	coefficient of *polynomial* and *sigmoid* kernels of SVM
ens	list of ensembles from all optimization steps
eta	the parameter of the *NSGA-II* algorithm
eta_c	the parameter of the *NSGA-II* algorithm for crossover
eta_m	the parameter of the *NSGA-II* algorithm for mutation
F_β	F-score classification metric with parameter β
F_i	ith fitness function
G	support function
G_{max}	number of generations

γ	kernel coefficient of SVM
h	degree (dimension of the new feature space)
i	class label
K	feature map
$\mathcal{LS}$	learning set
M	number of objectives
m_i	ith model
$\mathcal{M}$	set of labels (classes)
MB	the list of BAC metrics of models
MS	the list MB sorted descending
n	ensemble size
N	number of examples, dataset's size
Π	pool of base classifiers
Ψ	classification algorithm
p	pruning parameter
P_t	the t-th generation of the population
$prev$	list of models from previous iteration
Q	parameter of Q-statistic, pairwise diversity measure
Q_t	the t-th generation of the offspring population
r	root of center-based bootstrapping
R	results from criteria metrics
R^+	the sum of ranks where the second algorithm won in the Wilcoxon signed ranks test
R^-	the sum of ranks where the first algorithm won in the Wilcoxon signed ranks test
S_i	ith bootstrapped subset
T	neighborhood size
U	population size

V	a single feature vector
v	the number of votes for specific class
W	number of iterations of center-based bootstrapping
x	feature vector
x_i	ith example
$\hat{x}$	selected subset of features
$\overline{x}$	center of center-based bootstrapping
$\mathcal{X}$	feature space
y	class label vector

Abstract

The thesis focused on the use of multi-objective optimization for employing feature selection to classify mainly imbalanced data. Feature selection determines a feature subspace for each model, and this mechanism ensures the diversity of models in the ensemble. Research in this area has shown that optimizing for feature selection gives good results, and the proposed methods sometimes outperform state-of-the-art methods. An additional advantage of the proposed algorithms using multi-objective optimization is the ability to select the most appropriate solution, which classical methods do not offer. The following research objectives confirmed the research hypothesis formulated at the beginning.

Using multi-objective optimization to train classifiers on feature subspaces produces models with no worse prediction quality than state-of-the-art methods and allows choosing a solution tailored to a user's needs.

Several objectives were formulated to prove the hypothesis.

- **Development of feature selection methods based on Multi-Objective Optimization for constructing single classifiers.**

  This objective was met by proposing methods using optimization to perform feature selection. The methods use optimization algorithms such as *GA* in the single-criteria version and *NSGA-II* in the multi-objective version. The advantage of using optimization is taking into account not only the quality of the built classifier but also the cost of features, which is particularly important in cost-sensitive learning. Simultaneous optimization of two criteria, maximizing performance and minimizing cost, in the case of multi-objective optimization, gives a set of solutions from which the user can choose the most suited to his needs. The proposed methods achieved comparable quality to classical methods, but the latter do not allow the possibility of choosing from a solution set and returning only one solution.

- **Development of a multi-objective method for training classifier ensembles using learning Support Vector Machines base classifiers on subspaces of the feature space.**

  This goal was achieved by proposing the *SEMOOS* method. *SEMOOS* is an ensemble consisting of single *SVM* classifiers using multi-objective optimization and the *NSGA-II* algorithm to search the feature space and find two parameters of *SVM* classifiers. *NSGA-II* returns a set of such solutions, and each of them is used to train a model, which is then added to the pool, creating an ensemble classifier. Using the proposed center-based bootstrapping and model pruning in the method is optional. The method was tested on many imbalanced datasets and achieved satisfactory results compared to reference methods.

- **Development of a method for training classifier ensemble using learning decision tree base classifiers on subspaces of the feature space and aggregated criteria.**

  The objective was completed by presenting the *DE-Forest* method using the *Differential Evolution* optimization algorithm to find the best feature vector for the entire ensemble relative to various aggregated metrics. Such a vector is appropriately prepared, and based on it, individual decision tree models can be trained to create an ensemble. The proposed method often outperforms state-of-the-art methods.

- **Development of a multi-objective method for training classifier ensemble using learning decision tree base classifiers on subspaces of the feature space to form the non-random forest.**

  The goal was achieved by proposing the *MOOforest* method, an ensemble built from individual decision trees using the *MOEA/D* multi-objective optimization algorithm. *MOEA/D* searches the feature space for the entire ensemble based on two criteria simultaneously: *Precision* and *Recall*. Thanks to this, it returns a set of solutions from which one solution is selected using the *PROMETHEE II* function. The models are trained based on this solution that makes up the final ensemble. The proposed method, in many cases, outperforms the reference methods.

- **Development of a multi-objective method for training classifier ensemble using learning base classifiers on subspaces of the feature space and local optimization.**

  The goal was achieved by proposing the *MOLO* method. It is a novel method using multi-objective local optimization to build a diverse ensemble. Each base *DT* model is trained on one bootstrapped subset. The optimization searches through possible

solutions in each step and adds one model to the ensemble. The restrictions prevent the search from spreading to a considerable extent, and thanks to it, the algorithm selects the paths that provide the best results at a given moment. The *MOLO* method has been tested for two sets of dual criteria and can also handle the three criteria case. Finally, *MOLO* returns several ensembles from which the user can choose the best one for his needs. Extensive tests for imbalanced data comparing the proposed and reference methods showed *MOLO* advantage.

- **Experimental evaluation of the obtained methods.**

  This objective was achieved by comparing all proposed methods: *SEMOOS*, *DE-Forest*, *MOOforest*, and *MOLO*. Statistical tests and detailed results for each dataset showed each method's characteristics and competence areas.

Streszczenie

W pracy skupiono się na zastosowaniu optymalizacji wielokryterialnej w zadaniu selekcji cech w klasyfikacji głównie danych niezbalansowanych. Selekcja cech wyznacza podprzestrzeń cech dla każdego modelu, a mechanizm ten zapewnia różnorodność modeli w zespole. Badania w tym obszarze wykazały, że optymalizacja pod kątem selekcji cech daje dobre rezultaty, a proponowane metody czasami przewyższają metody referencyjne. Dodatkową zaletą proponowanych algorytmów wykorzystujących optymalizację wielokryterialną jest możliwość wyboru najlepszego rozwiązania, czego nie oferują metody klasyczne. Przedstawione poniżej cele badawcze potwierdziły postawioną na wstępie hipotezę badawczą.

Zastosowanie optymalizacji wielokryterialnej do uczenia klasyfikatorów na podprzestrzeniach cech pozwala uzyskać modele o jakości predykcji nie gorszej niż metody referencyjne i pozwala na wybór rozwiązania dostosowanego do potrzeb użytkownika.

Aby udowodnić hipotezę, sformułowano kilka celów.

- **Opracowanie metod selekcji cech w oparciu o optymalizację wielokryterialną do konstruowania pojedynczych klasyfikatorów.**

  Cel ten został osiągnięty poprzez zaproponowanie metod wykorzystujących optymalizację do przeprowadzenia selekcji cech. Metody wykorzystują algorytmy optymalizacyjne takie jak *GA* w wersji jednokryterialnej i *NSGA-II* w wersji wielokryterialnej. Zaletą stosowania optymalizacji jest uwzględnienie nie tylko jakości zbudowanego klasyfikatora, ale także kosztu funkcji, co jest szczególnie ważne w uczeniu uwzględniającym koszty. Jednoczesna optymalizacja dwóch kryteriów, maksymalizacja wydajności i minimalizacja kosztów, w przypadku optymalizacji wielokryterialnej daje zestaw rozwiązań, spośród których użytkownik może wybrać najbardziej dopasowane do swoich potrzeb. Zaproponowane metody osiągnęły jakość porównywalną z metodami klasycznymi, przy czym te ostatnie nie pozwalają na możliwość wyboru ze zbioru rozwiązań i zwracają tylko jedno rozwiązanie.

- **Opracowanie wielokryterialnej metody uczenia zespołów klasyfikatorów z wykorzystaniem uczenia maszyn wektorów nośnych (Support Vector Machines), które opierają klasyfikatory na podprzestrzeniach przestrzeni cech.**

  Cel ten osiągnięto proponując metodę *SEMOOS*. *SEMOOS* jest zespołem składającym się z pojedynczych klasyfikatorów *SVM* wykorzystujących optymalizację wielokryterialną i algorytm *NSGA-II* do przeszukiwania przestrzeni cech i znajdowania dwóch parametrów klasyfikatorów *SVM*. *NSGA-II* zwraca zestaw takich rozwiązań, a każde z nich służy do uczenia modelu, który następnie jest dodawany do puli, tworząc klasyfikator zespołowy. Stosowanie w tej metodzie proponowanego losowania zbioru treningowego i testowego (center-based bootstrapping) oraz usuwanie modeli z finalnego zespołu klasyfikatorów (pruning) jest opcjonalne. Metoda została przetestowana na wielu niezbalansowanych zbiorach danych i uzyskała zadowalające wyniki w porównaniu z metodami referencyjnymi.

- **Opracowanie metody uczenia zespołu klasyfikatorów z wykorzystaniem klasyfikatorów bazowych drzew decyzyjnych na podprzestrzeniach przestrzeni cech i zagregowanych kryteriów.**

  Cel został zrealizowany poprzez zaprezentowanie metody *DE-Forest* wykorzystującej algorytm optymalizacyjny *Differential Evolution* w celu znalezienia najlepszego wektora cech dla całego zespołu w odniesieniu do różnych zagregowanych metryk. Taki wektor jest odpowiednio przygotowany i na jego podstawie można wytrenować poszczególne modele drzew decyzyjnych, aby utworzyły zespół. Zaproponowana metoda często przewyższa metody referencyjne.

- **Opracowanie wielokryterialnej metody uczenia zespołu klasyfikatorów z wykorzystaniem klasyfikatorów bazowych drzew decyzyjnych na podprzestrzeniach przestrzeni cech w celu utworzenia lasu nielosowego.**

  Cel ten osiągnięto proponując metodę *MOOforest*, czyli zespół zbudowany z poszczególnych drzew decyzyjnych przy użyciu algorytmu optymalizacji wielokryterialnej *MOEA/D*. *MOEA/D* przeszukuje przestrzeń cech dla całego zestawu w oparciu o jednocześnie dwa kryteria: *Precision* i *Recall*. Dzięki temu zwraca zbiór rozwiązań, z których przy pomocy funkcji *PROMETHEE II* wybierane jest jedno rozwiązanie. Modele są trenowane w oparciu o to rozwiązanie, które składa się na finalny zespół. Zaproponowana metoda w wielu przypadkach przewyższa metody referencyjne.

- **Opracowanie wielokryterialnej metody uczenia zespołu klasyfikatorów z wykorzystaniem klasyfikatorów bazowych na podprzestrzeniach przestrzeni cech i optymalizacji lokalnej.**

Cel został osiągnięty poprzez zaproponowanie metody *MOLO*. Jest to oryginalna metoda wykorzystująca wielokryterialną optymalizację lokalną w celu zbudowania zróżnicowanego zespołu. Każdy podstawowy model *DT* jest trenowany na jednym wylosowanym podzbiorze. Optymalizacja przeszukuje możliwe rozwiązania na każdym etapie i dodaje jeden model do zespołu. Ograniczenia w znacznym stopniu zapobiegają rozprzestrzenianiu się poszukiwań, dzięki czemu algorytm wybiera ścieżki, które w danym momencie zapewniają najlepsze wyniki. Metodę *MOLO* przetestowano dla dwóch zestawów podwójnych kryteriów i może ona również obsłużyć przypadek trzech kryteriów. Na koniec *MOLO* zwraca kilka zestawów, spośród których użytkownik może wybrać ten, który najlepiej odpowiada jego potrzebom. Obszerne testy na danych niezbalansowanych, porównujące metody proponowane i referencyjne, wykazały przewagę metody *MOLO*.

- **Eksperymentalna ocena otrzymanych metod.**

  Cel ten osiągnięto poprzez porównanie wszystkich zaproponowanych metod: *SE-MOOS*, *DE-Forest*, *MOOforest* i *MOLO*. Testy statystyczne i szczegółowe wyniki dla każdego zestawu danych wykazały charakterystykę każdej metody i obszary ich kompetencji.

Contents

Acknowledgements **iv**

Abbreviations **v**

Symbols **ix**

1 Introduction **11**

2 Related works **15**
2.1 Machine learning 15
2.2 Classification 16
2.3 Multi-objective optimization 30
2.4 Multi-objective optimization in ensemble learning 35

3 Feature selection method **39**
3.1 Experimental evaluation 42
3.2 Results 45
3.3 Lessons learned 48

4 SVM Ensemble with Multi-Objective Optimization Selection **51**
4.1 Algorithm 52
4.2 Experimental evaluation 57
4.3 Experiments 59
4.4 Lessons learned 67

5 Ensemble learning on feature subspace methods **71**
5.1 DE-Forest – optimized decision tree ensemble 71
5.2 MOOforest – multi-objective optimization to form decision tree ensemble . 83

6 Multi Objective Local Optimization Forest **93**
6.1 Experimental evaluation 98
6.2 Results 101
6.3 Lesson learned 103

7 Comparison of proposed algorithms **105**

8 Conclusion and future research directions **113**

Bibliography **119**

Chapter 1

Introduction

The industrial revolution [65] has been changing the world since the 18th century. Subsequent industrial revolutions include the invention of the steam engine, the introduction of electricity and production lines in the 19th century, and the development of information and communication technologies in the 20th century. In the 21st century, we are in the middle of an Artificial Intelligence (AI) [73] revolution. Each revolution changed the way of production, trade, and work and led to societal changes. Companies and countries that are the most technologically advanced tend to develop faster. Like any revolution, AI raises particular societal concerns, such as the loss of employment or even the domination of machines. AI is already being used in many fields on an increasingly broader scale. Examples include chatbots [8] using AI to communicate with customers of online stores, ChatGPT [131] that conducts a conversation with the user on various topics, or medical systems allowing for faster and cheaper diagnosis [101].

AI is a general term that includes subcategories such as Natural Language Processing (NLP) [30], Image Processing [167], and Machine Learning (ML) [170]. The last one includes, among others, classification, in which an effective model training process leads to the assignment of appropriate labels to the data. Classification is one of the main topic of this dissertation.

The industrial revolution also led to the optimization of production processes. Such processes have many criteria, so defining appropriate optimization objectives can be problematic. Hence, a natural solution is to use a more significant number of criteria, the simultaneous optimization of which leads to obtaining the best possible results tailored to a given problem. This approach is called multi-objective optimization (MOO) [42]. An example would be a production line implementation, which has increased factories' efficiency with a shorter time to produce an item, minimized losses, and improved financial management. Such optimization task has three criteria. When an algorithm

optimizes a problem based on multiple criteria, it returns a set of best solutions instead of a single one. Then, depending on the user's priorities, the final solution is more tailored to his needs. The user can choose which criterion is most important and choose the solution that achieves a better result for this particular criterion.

A wide selection of solutions is an advantage of multi-objective algorithms compared to single-objective ones. Single-criteria algorithms often use aggregated criteria. However, this does not produce the same effect as simultaneous optimization of two or more, often contradictory, criteria. The aggregated criterion does not provide sufficient control over the optimization process. It has little information because the aggregated criterion cannot be returned to the individual values of the components [69].

On the other hand, the industrial revolution also means greater exploitation of the natural environment – fuel and electricity, often from non-renewable energy sources, power all machines use daily. Environment protection is becoming an increasingly important topic because we can already see the harmful effects of progress. However, in order not to thoroughly criticize progress and research into new technologies, it is worth mentioning *Green AI* [126], which decreases *AI*'s carbon footprint. Deep learning or training *NLP* models consume many resources, and researchers strive for the best possible performance result. However, *Green AI* is gaining popularity, and more attention is being paid to its environmental aspects. Researchers must consider many criteria, not only the performance of their methods but also the cost and efficiency.

The cost criterion is easy to visualize in medicine [127]. Data is collected from patients differently, and each technique has an associated cost. One of the cheapest ways to collect data is to interview the patient. This data may include personal information such as age, weight, type of work, or information about pain that affects the diagnosis. However, various medical tests give more specific information, and the cost of the tests must be considered. The more complicated the test or the more advanced the tools, the more expensive the test is. All the information mentioned is treated as features of a dataset containing data from many patients, i.e., many examples. Each feature may have an assigned cost to obtain it.

Cost can also be understood as the cost of incurring an incorrect decision. The system classifies the patient whether he or she has cancer; hence, two cases can be indicated. The patient is healthy, and the system determines that the patient has cancer. In the second case, the patient is sick, and the system says he or she is healthy. The costs of both errors may have different values, but it is challenging to provide a specific value for a given error in this case. It can be concluded that the latter error has much more significant consequences because the disease is not detected, and the cost of such a decision is very high for the patient [157].

Considering the aspects mentioned above, the thesis focuses on using multi-objective optimization (*MOO*) algorithms, which provide greater possibilities for the user's choice of solution than single-criteria algorithms. Some proposed approaches also use methods to select a solution automatically but still consider the user's needs. This work uses *MOO* algorithms to select features in the classification task by individual and classifiers ensemble. A research hypothesis is formulated as follows:

Using multi-objective optimization to train classifiers on feature subspaces produces models with no worse prediction quality than state-of-the-art methods and allows choosing a solution tailored to a user's needs.

In order to verify the above research hypothesis, the following research objectives were proposed:

- Developing feature selection methods based on multi-objective optimization for constructing single classifiers.
- Proposing a multi-objective method for building classifier ensembles using learning Support Vector Machines trained on subspaces of the feature space.
- Developing a classifier ensemble forming algorithm using decision tree classifiers trained on subspaces of the feature space and aggregated criteria.
- Developing a multi-objective method using learning decision trees on subspaces of the feature space to form the non-random forest.
- Employing a multi-objective optimization to form a classifier ensemble using classifiers trained on the feature space subspaces and local optimization.
- Experimental evaluation of the developed algorithms.

The structure of the thesis is as follows. Chapter 2 introduces the concepts and literature review related to the topic of the work, starting with machine learning and classification and ending with the equally extensive topic of multi-objective optimization. Chapter 3 proposes and compares various feature selection techniques, including single- and multi-objective genetic algorithms, and then classifies multi-class data. Chapter 4 presents the *SEMOOS* algorithm – an approach based on multi-objective optimization which returns a pool of non-dominated solutions. Chapter 5 shows two classifiers ensemble built on the decision tree and feature subspaces found by optimization algorithms. The first classifier uses *Differential Evolution* as a single-criteria algorithm (Sec. 5.1), and the second uses *MOEA/D* as a multi-objective algorithm (Sec. 5.2). Chapter 6 describes the *MOLO* method, which employs the local optimization algorithm to add different models to the

ensemble, thus creating the ensemble that achieves the best classification quality at a given moment. Chapter 7 compares all the proposed methods and performs a statistical analysis of selected datasets. Chapter 8 summarizes the entire work and indicates future research directions.

Chapter 2

Related works

This chapter introduces the reader to the dissertation topic. It covers the subjects of machine learning and classification, detailing imbalanced data classification. The work describes various classifiers, including ensembles and preprocessing methods, such as feature selection and extraction. A vital element of the dissertation is multi-objective optimization. The chapter ends with a section on using multi-objective optimization in ensemble learning.

2.1 Machine learning

Machine learning, derived from artificial intelligence, teaches a computer how to perform a task without programming it specifically for that. The computer receives input data and, through experience, acquires knowledge that allows it to solve various problems. The machine learning process is very similar to the human learning. Arthur Samuel coined the term machine learning in 1959, and since then, there have been many definitions of the concept [170].

Machine learning is divided into four categories (Fig. 2.1): supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. If the algorithm receives labeled training data, i.e., each sample is assigned a class, it is supervised learning. Suppose it is unlabeled data – unsupervised learning. A combination of the two approaches mentioned above is semi-supervised learning, which usually has a more extensive set of unlabeled data and uses knowledge from a small amount of labeled data. Reinforcement learning is a slightly different case because these algorithms do not use training data directly but rather the environment from which they automatically download data. The agent performs interactions in the environment using policy to obtain the maximum reward [137].

Dimensionality reduction and clustering fall into the category of unsupervised learning. The former uses algorithms to reduce the number of features and leave only features with important information. The latter aims to group instances with similar characteristics into categories, thus creating subsets that represent the population [45].

Supervised learning algorithms try to find the relationship between the input and output of the learning process. The model receives prepared features and tries to predict target output based on them. In regression, the answer is continuous, while in classification, the answer is discrete, i.e., class labels.

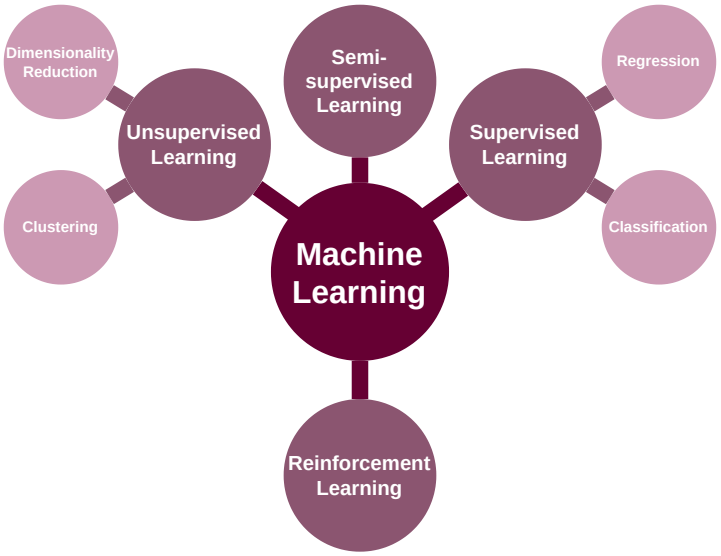


Figure 2.1: *Taxonomy of Machine Learning*

Since the proposed algorithms classify data, this subject will be described in more detail in the next section.

2.2 Classification

Classifiers are algorithms that perform classification on data and return the answer in the form of a class label. The model receives properly prepared training data with a class assigned to each instance. Then, it learns from this data which label (class) corresponds to which feature values. After this, it can classify an unseen incoming sample.

Let us present the mathematical notation of **classification** [5, 153]. Let $\mathcal{X}$ denotes a feature space, and d a number of features, so a feature vector x is

$$x = \left[x^{(1)}, x^{(2)}, \ldots, x^{(d)}\right]^T, \text{ and } x \in \mathcal{X} = \mathcal{X}^{(1)} \times \mathcal{X}^{(2)} \times \cdots \times \mathcal{X}^{(d)} \tag{2.1}$$

The distance between two examples is an attribute, which can be numerical or categorical. The first category is divided into discrete values (e.g., number of visits) and continuous values (e.g., height in cm). The categorical contains the following elements: nominal (unorder, e.g., male/female) and ordinal (ordered, e.g., low/medium/high). When using categorical attributes, an appropriate data encoding technique must be used to convert the categories to numbers because most machine learning models only consider numerical values. The most commonly used encoders are label encoder, one-hot encoder, or dummy encoder. We may assume that attributes belong to a set of real numbers $\mathbf{R}$ [33]

$$x \in \mathcal{X} \subseteq \mathbf{R}^d \tag{2.2}$$

The main task of the classification algorithm Ψ is to assign a label from the set of labels (classes) $\mathcal{M}$ to a new sample that enters the system using its feature vector x. Suppose $\mathcal{X}$ is a domain and $\mathcal{M}$ is a codomain. The function Ψ is

$$\Psi : \mathcal{X} \to \mathcal{M} \tag{2.3}$$

The classifier makes a decision using so-called support functions $(G_1, G_2, ..., G_i)$ that returns support for the corresponding class [153]. Then, the maximum rule is used

$$\Psi(x) = \max_{k \in \mathcal{M}}(G_k(x)) \tag{2.4}$$

Fig. 2.2 shows the classification pipeline, from obtaining raw data to output as the classifier's performance. In the first phase – preprocessing, the input data is annotated. It is an essential task at the data preparation stage, where errors that affect further classification may appear. The following process is data preparation, including cleaning them, fixing errors, and filling empty gaps. Dimensionality reduction mechanisms, such as feature extraction and selection, are often used when the data contains many features. Then, a model must receive the prepared dataset, so the dataset is divided into train and test data, which makes it possible to assess the quality of the model. The train data is used during learning, and this data is denoted as learning set $\mathcal{LS}$, where x_N is Nth sample, and i_N is a label of Nth sample, N is the size of the dataset given at the beginning

$$\mathcal{LS} = \{(x_1, i_1), (x_2, i_2), \ldots, (x_N, i_N)\} \tag{2.5}$$

The second phase is learning. The classifier is the output of the selected machine learning algorithm Ψ, and it receives training data. It learns what feature values cause selecting one of the available labels. The final phase is algorithm evaluation. This time, the trained algorithm receives test data and makes predictions. Then, the classifier checks its responses with actual labels. On this basis, the program calculates metric performance that indicates how well the classifier performed its task.

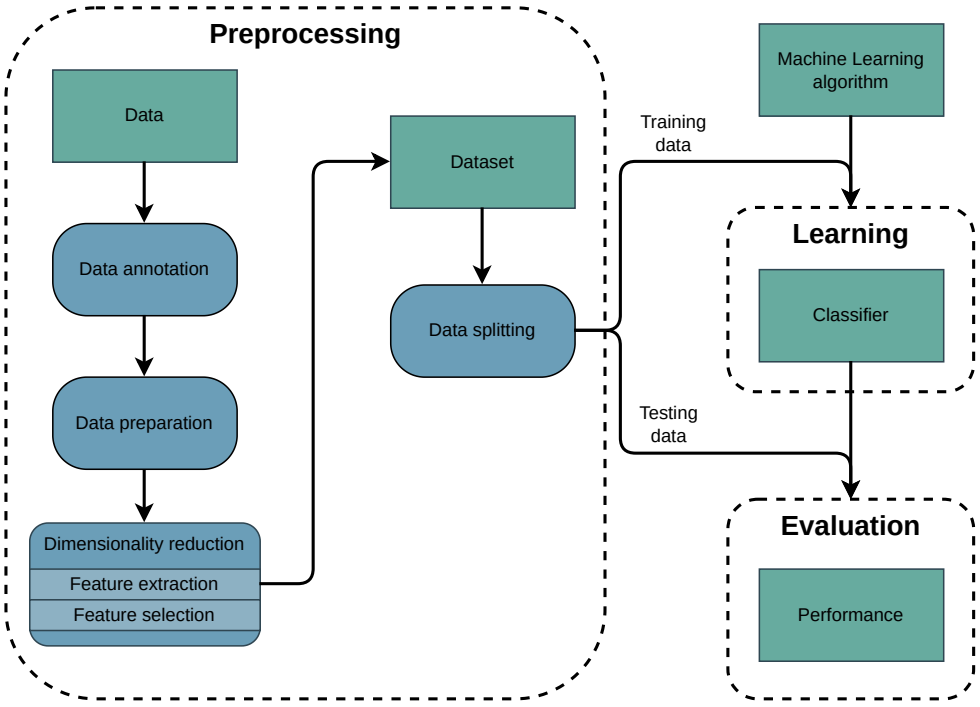


Figure 2.2: *Classifier learning process*

The classifier's performance must be measured to choose the best classifier for a given problem. In the case of binary data, where we only have positive and negative classes, a confusion matrix can be used (Fig. 2.3). This matrix compares the actual values with the values predicted by the algorithm. Values marked as *True Positive* (TP) and *True Negative* (TN) are correctly recognized classes. *False Positive* (FP) and *False Negative* (FN) indicate incorrect classification. FP is when the classifier recognizes the negative class as a positive, FN otherwise.

Confusion matrix is the basis for determining frequently used metrics: *Accuracy*, *Recall*, *Specificity*, *Precision*.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.6}$$

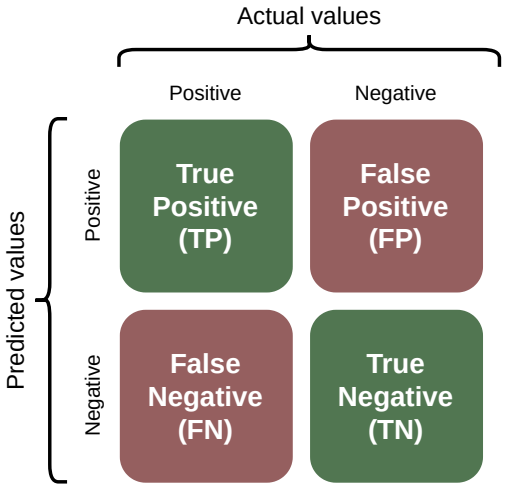


Figure 2.3: *Confusion matrix for two class problem*

$$Recall = \frac{TP}{TP + FN} \tag{2.7}$$

$$Specificity = \frac{TN}{TN + FP} \tag{2.8}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.9}$$

Based on the primary metrics, aggregated metrics are also listed: *Geometric mean* based on *Precision* and *Recall* ($Gmean$), *Geometric mean* based on *Specificity* and *Recall* ($Gmean_s$), *False Positive Rate* (FPR), *False Negative Rate* (FNR), F_β, $F_1 score$, and BAC.

$$Gmean = \sqrt{Precision \times Recall} \tag{2.10}$$

$$Gmean_s = \sqrt{Specificity \times Recall} \tag{2.11}$$

$$FPR = 1 - Specificity = \frac{FP}{TN + FP} \tag{2.12}$$

$$FNR = 1 - Recall = \frac{FN}{TP + FN} \tag{2.13}$$

$$F_\beta = (1+\beta^2)\frac{Precision \times Recall}{(\beta^2 \times Precision) + Recall} \tag{2.14}$$

$F_1 score$ is a particular case of the F_β metric where $\beta = 1$:

$$F_1 score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.15}$$

The *Balanced Accuracy* (BAC) metric is used to classify imbalanced data because regular *Accuracy* gives misleading results in this case

$$BAC = \frac{Recall + Specificity}{2} \tag{2.16}$$

All metrics and formulas described above can only be used for two-class problems. In the case of multi-class datasets, it is also possible to use some metrics, but then the *Micro-averaging* or *Macro-averaging* mechanism must be used. The *Micro* strategy favors classes that occur more frequently, while the *Macro* strategy averages the results across all classes. Branco et al. [16] collected metrics equations used in multi-class classification and grouped them into *Recall*-based, *Precision*-based, and general metrics.

After the theoretical description of the classification, we present examples of specific models. Only selected basic classifiers that were used in the thesis will be described.

Vapnik proposed the **Support Vector Machine** (SVM) [144], and Burges presented a paper devoted to a detailed description of this method for the pattern recognition task [23]. Currently, SVM is mainly used for classification but can also be used for regression or outlier detection tasks. The main goal of SVM is to find the optimal hyperplane, i.e., the decision boundary between data points from different classes in the feature space. Support vectors are the closest points to the hyperplane; their distance is called the margin. The algorithm is designed to maximize margin, which leads to better classification performance. One of the essential parameters of SVM is the kernel. It is a mathematical function that maps the original data points into high-dimensional feature space, making the kernel work well for nonlinear-separable problems. There are several kernel functions: *linear*, *polynomial*, *radial basis function* (RBF), and *sigmoid* [153]. The kernel functions are defined as the dot product of vectors mapped by feature map $K(x_i, x_j)$, where x_i and x_j are the following examples from the learning set $\mathcal{LS}$, h is the degree (dimension of the new feature space), γ is the kernel coefficient parameter ($\gamma = \frac{1}{2\sigma^s}$), and e is a coefficient of *polynomial* and *sigmoid* [125]:

- $K(x_i, x_j) = \langle x_i, x_j \rangle$ – *linear*

- $K(x_i, x_j) = (\gamma\langle x_i, x_j\rangle + e)^h$ – *polynomial*
- $K(x_i, x_j) = \exp(-\gamma||x_i - x_j||)^2$ – *RBF*
- $K(x_i, x_j) = \tanh(\gamma\langle x_i, x_j\rangle + e)$ – *sigmoid*

Two parameters γ and C used in kernel functions are essential for SVM performance. The γ parameter determines how much influence a single training example has on a function's performance. The larger γ, the closer the other examples must be for the effect to be significant. The C parameter is a penalty or regulation parameter. It determines the trade-off between minimizing error and maximizing the classification margin. Changing the value of C controls the training and testing error, number of support vectors, and SVM margin [138].

Decision tree (DT) is a non-parametric method for classification and regression. It is a hierarchical model of many decision nodes containing tests of attribute values and final leaves having the algorithm's output. The decision-making process starts at the root, goes through branches, and to another node, each time implementing a test function. Once the algorithm reaches a leaf, the value stored there is the DT's response. The DT divides the problem into smaller sub-problems until it can select the final class. One of the main advantages of a tree is its interpretability. A tree's clear structure can be turned into a set of if-then rules [5].

Many individual classifiers connected by a combination rule create **ensemble classifiers**, also known as multiple classifier systems or committees. Fig. 2.4 presents creating the ensemble classifier. The classifier pool is

$$\Pi = \{\Psi_1, \Psi_2, \Psi_3, \ldots, \Psi_n\} \tag{2.17}$$

Ensemble methods train base learners on training data. Examples of base learners might be DT or SVM. The base learners form a homogeneous ensemble if they are the same type. Suppose they are different, then the ensemble is heterogeneous. Base learners can be imperfect classifiers because the combination of even weak classifiers results in a good ensemble. Each base learner has an area of competence where it achieves the best classification. Thanks to this, when combined, they create a diverse ensemble [82].

Unfortunately, achieving diversity in an ensemble is not easy, and the concept of diversity needs to be formulated better. Hence, there are many diversity measures, but none are universal. Kuncheva and Whitaker [83] tested ten diversity measures, including *Q statistics*, the *disagreement measure*, and *Kohavi-Wolpert* variance. The authors' assumptions

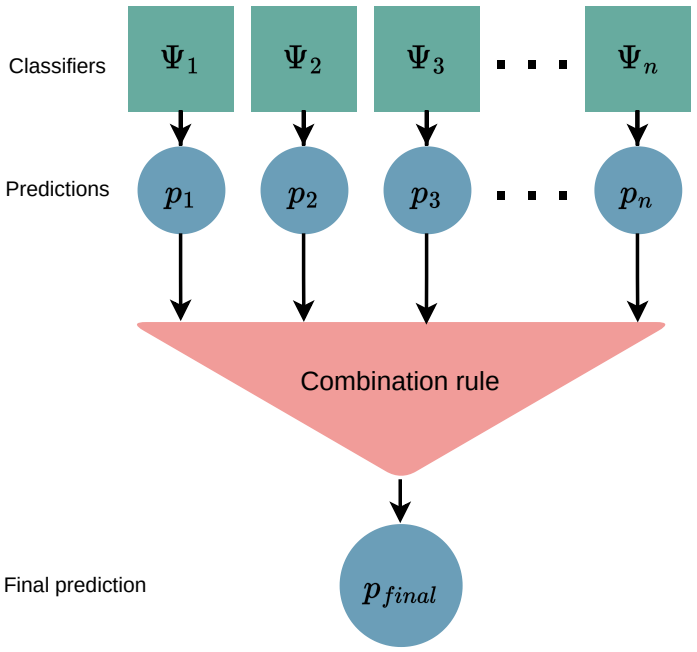


Figure 2.4: *Ensemble classifier*

about the correlation between already proposed diversity measures and average *Accuracy* did not come true in every case, which was a surprising conclusion.

Ensembles can have different combination rules, the most famous of which fall into the *Aggregating* or *Voting* categories. *Aggregating* involves collecting outputs from base learners and averaging them. This group includes *Simple Averaging* and *Weighted Averaging*. *Voting* collects votes from each individual classifier in the form of an exact class value or class probability. The second category, *Voting* rule, is divided into *Majority Voting*, *Plurality Voting*, *Weighted Voting*, and *Soft Voting*. There are also many other rules, such as *Stacking* and *Algebraic* methods. Depending on the selected combination rule, the ensemble classifier collects responses from base learners and makes the final decision on the class to which to assign each data sample [169].

An essential aspect of building ensembles is not only adding models but also removing them or selecting them from the pool, i.e., pruning. It leads to a reduction in the number of models in the ensemble and improves predictive performance. Typically, pruning is based on a metric, such as performance, diversity, or margin. Rankings can be determined based on the metrics, and then it is rank-based pruning, where each model in the ensemble receives its rank. Ensemble selects only the highest-ranking models, thus creating the best possible ensemble for the selected metrics. Models get higher ranks when the metric achieves more satisfactory values; this can be a minimum or maximum, depending on the metric. Mohammed et al. [104] present order-based pruning metrics.

An example of an ensemble classifier is a **Random Forest** (*RF*). *RF* is a method built of decision trees for classification and regression tasks. The algorithm constructs each tree on a random subset containing different features. The *RF* then collects the responses of all the trees and uses *Majority Voting* or *Averaging* to make the final decision. Each tree is different, providing a larger coverage area and ensuring diversity [19].

2.2.1 Imbalanced data

When the number of instances in one class is significantly lower than in the others – dataset is **imbalanced**. For the binary task, a more numerous class is called the majority (negative) class, while the other is the minority (positive) [97]. Because the consequences of making a mistake might be pretty substantial, special care is used while classifying positive cases [43].

In imbalanced data classification, the disproportion among the different classes is not the sole issue of learning difficulties. One may quickly devise an example where the instance distributions from different classes are well-separated. Napierała and Stefanowski observed that the minority class samples often may form scattered clusters of an unknown structure [106]. An additional complication arises from the possibility that there may be an insufficient number of minority class samples for a classifier learning algorithm to achieve an adequate level of generalization, resulting in *overfitting* [29].

One may divide imbalanced data classification algorithms into three groups [98].

Data preprocessing methods concentrate on decreasing the number of examples from the majority class (*undersampling*) or generating new minority class samples (*oversampling*). These mechanisms aim to balance the number of objects from considered classes. *Oversampling* randomly replicates existing samples or generates new samples in a guided manner. *SMOTE* is the most recognized algorithm [27] that generates new minority samples between two randomly selected objects. Unfortunately, methods such as *SMOTE* may change the characteristic of the minority class and, as a result, *overfit* the classifier. Therefore, several modifications of *SMOTE* have been proposed that can identify the samples to be copied more intelligently, such as *Borderline SMOTE* [62] that generates new minority class samples close to the decision border. *Safe-Level SMOTE* [22] and *LN-SMOTE* [100] reduce the probability of generating synthetic minority instances in areas where the predominant objects are that of the majority class. Koziarski et al. proposed *RBO* [78] and *CCR* that enforce instances from the majority-class to be relocated from the areas where the minority-class instances are present [79]. The alternative preprocessing approach is *undersampling*. Such methods employ randomly removing the

instances from the majority class or removing them from the areas so that the classifier's quality is not disrupted using neighborhood analysis [98].

Inbuilt mechanisms modify existing classification algorithms for imbalanced tasks, focusing on predictive performance for both classes. One approach is one-class classification [71], which aims to learn the decision areas associated with one class. Initially, an approach based on building models for the majority class was proposed due to the sufficiently large number of objects representing it, and the minority class was treated as so-called outliers. Krawczyk et al. proposed a different approach in [80], where a one-class classifier was trained on a minority class. In turn, *cost-sensitive* classification considers the asymmetrical loss function that assigns a higher misclassification cost of minority class [63, 81, 98, 171].

Hybrid methods combine the advantages of methods using data preprocessing with the different classification methods. One of the most popular approaches is the hybridization of *under-* and *oversampling* with ensemble classifiers [52]. This approach allows the data to be independently processed for each base model. It is also worth noting methods based on ensemble classification [152], such as *SMOTE Boost* [28] and *AdaBoost.NC* [147].

Classification of imbalanced data can also be treated as an optimization task, for which, among others, **Evolutionary Algorithms** (EA) are used. *EA* includes, among others, genetic algorithms, swarm intelligence methods, and multi-objective optimization algorithms. Pei et al. [115] presented a survey describing many *EA* methods for classifying imbalanced data. Because *EA*s have good global search ability, they are suitable for data sampling, especially undersampling. Garcia and Herrera [54] proposed eight *GA*-based evolutionary undersampling methods that ensure a good trade-off between the balance of distribution of classes and the performance of the methods. In [85], the proposed clustering method reduces computational costs, which is a particular problem in the case of large-scale datasets. *EA* is also used in cost-sensitive learning. In [87], a method based on multi-objective optimization provides a trade-off between the generalization ability and the case-weighting factors using a cost matrix.

The imbalanced data classification cannot be based on the *Accuracy* metric because it does not indicate the correct results and biases the model towards the majority class. *Accuracy* may be high due to the proper classification of the majority class, but then the minority class is omitted [21]. Therefore, choosing the right metrics to classify imbalanced data is crucial. Basic metrics, such as *Precision* and *Recall*, considering only one class at a time, are good choices. However, sometimes classifiers require aggregate metrics that return their performance. Instead of using *Accuracy*, *Balanced Accuracy* or *Gmean* is a more practical alternative [60].

2.2.2 Preprocessing

Preprocessing is a crucial step in the classification process. Preprocessing techniques prepare data before passing it to the algorithm, which makes it more effective. Data can contain an extensive number of features, which is called *Bellman's concept – the curse of dimensionality* [10]. Additionally, a more significant number of features does not guarantee better classification because some features may be non-informative. Even increasing features may lead to a decreasing classifier performance, assuming an unchanged number of observations, which is called the *Hughes effect* [66]. In this case, the classifier's performance could only be improved if the training set's size increased with the number of features. It is a challenging task, especially in the case of real data, so the aim is to reduce dimensionality, thus excluding irrelevant and redundant data, which ensures better classification, faster calculations, and a better understanding of the data [163].

One of the basic dimensionality reduction techniques is **feature selection** (*FS*). *FS* selects some of the features from the original dataset and creates a new feature set. Using feature selection gives better classification performance than the performance on the complete set of features. Cai et al. [24] presented the division of feature selection methods into five categories, referring to the following aspects:

1. Labeled, unlabeled, or partially labeled training data – supervised, unsupervised, and semi-supervised feature selection methods.
2. Relationship with learning methods – filter, wrapper, and embedded methods.
3. Evaluation criterion – correlation, Euclidean distance, consistency, dependence, and information measure criteria.
4. Search strategies – forward increase, backward deletion, random, and hybrid models.
5. The type of the output – feature rank (weighting) and subset selection methods.

Let us focus on 2 because we use these methods for feature selection later in the work. Filter methods use statistical measures to select features, such as *Information gain*, *Chi-square*, *Fisher score*, *correlation coefficient*, and *variance threshold* [145]. Ikram and Cherukuri mentioned that *Chi-square* is the best method for multi-class problems [135]. The *Chi-square* test statistic applied to the *Select K-best* function chooses K-best features from the datasets that are the most relevant to the classification process.

Feature selection depends on a classifier in wrapper methods. The classifier may use optimization algorithms. In this case, the population consists of vectors with binary-encoded

features. Many papers address the feature selection problem using one-objective optimization such as *Genetic Algorithms* (*GA*) [44]. *GA* searches a population, and through iteration, it evaluates and uses genetic operators (*selection*, *mutation*, *crossover*) to find the best solutions [41]. As confirmed by the survey [40], many types of metaheuristics, such as multi-objective ones, can be used in the feature selection task. Some *wrapper* methods make selections based on analyzing the results of a specific classifier [15, 117].

Embedded methods use ensemble classifiers for feature selection. An example of such a classifier is *Random Forest*, which consists of many decision-making models [19]. Also, some statistical methods are applied to select features: *ANOVA* [15], i.e., the analysis of the variance and *Pearson's correlation coefficient* [24, 119] which are better at a single criterion task.

Feature extraction is the second dimensionality reduction technique. It involves transforming features from the original set into a new set. Transformation reduces the size of the feature vector [163]. Unfortunately, unlike feature selection, extraction, by transforming and merging certain features, loses the ability to reproduce them. This process is irreversible and may result in loss of interpretability of the classifier's response. These days, it is undesirable because explainability is valuable in detection systems.

Despite this, many feature extraction methods have been developed, and the most popular of them are *Principal Component Analysis* (*PCA*) [141], *Linear Discriminant Analysis* (*LDA*), *Multi-Dimensional Scaling* (*MDS*), *Isometric Mapping* (*ISOMAP*), *Locally Linear Embedding* (*LLE*) and many more [7]. Methods based on deep learning are increasingly being used, where deep learning means that the system learns from experience and uses a hierarchy of concepts [55].

2.2.3 Evaluation

A new method must be well-tested to demonstrate its usefulness among other methods. We strive to ensure the algorithm has the lowest possible classification error concerning the relevant metrics. However, according to *Wolpert's "no free lunch"* theory, there are no ideal methods, and each has its area of competence. Therefore, a lot depends on data.

Selection of an appropriately diverse dataset or focus on one problem, if the purpose of the experiments is to find a method for a specific dataset, is vital. The learning set $\mathcal{LS}$ should be divided into a training set (*TS*) and a validation set (*VS*). *TS* is used to train the classifier, and *VS* is used to test it. The most basic method is hold-out; for example, 70% or 80% is allocated to *TS* and the remaining 30% or 20% to *VS*.

However, there are better methods than this because some problems may arise when validation is run only once. Some datasets are small, and when divided into two sets, they contain too few examples, or the examples are outliers that do not represent the learning set well. Generalization may occur, and the measured metrics will not indicate correct values.

Therefore, there must be at least several replications. One of the methods is *cross-validation*. The *TS* and *VS* sets must contain an appropriate number of examples. It is possible to divide the original dataset into k parts for large datasets and then assign them to *TS* and *VS*. However, dividing the same dataset by k times is usually used.

One of the popular approaches is ***k-fold cross-validation***, where the size of k is inversely proportional to the size of $\mathcal{LS}$. The larger $\mathcal{LS}$, the smaller k, usually k equals 10 or 30. Fig. 2.5 shows a *10-fold cross-validation*, determining one testing part and nine training parts from a dataset divided into ten sets. In subsequent iterations, the testing part contains further subsets, i.e., 1st iteration – first subset, 2nd iteration – second subset, etc. The remaining subsets from the dataset are allocated to the training part. Each iteration returns the performance, and at the end, it is averaged.

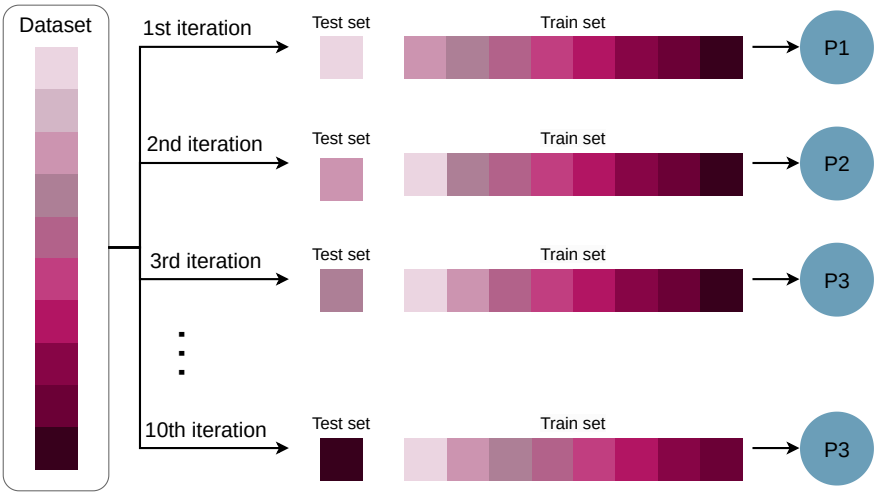


Figure 2.5: *k-fold cross validation*

If validation is repeated several times, we have Repeated Stratified K-Fold, for example, **5-splits × 2-fold cross-validation** – 5×2 *CV* (Fig. 2.6). In each iteration, the algorithm shuffles the dataset, learns on the first part of the dataset, and tests on the second part. In the next step, each iteration gives two performance results. This mechanism is repeated, so we obtain ten performance measures because we have ten folds (5×2).

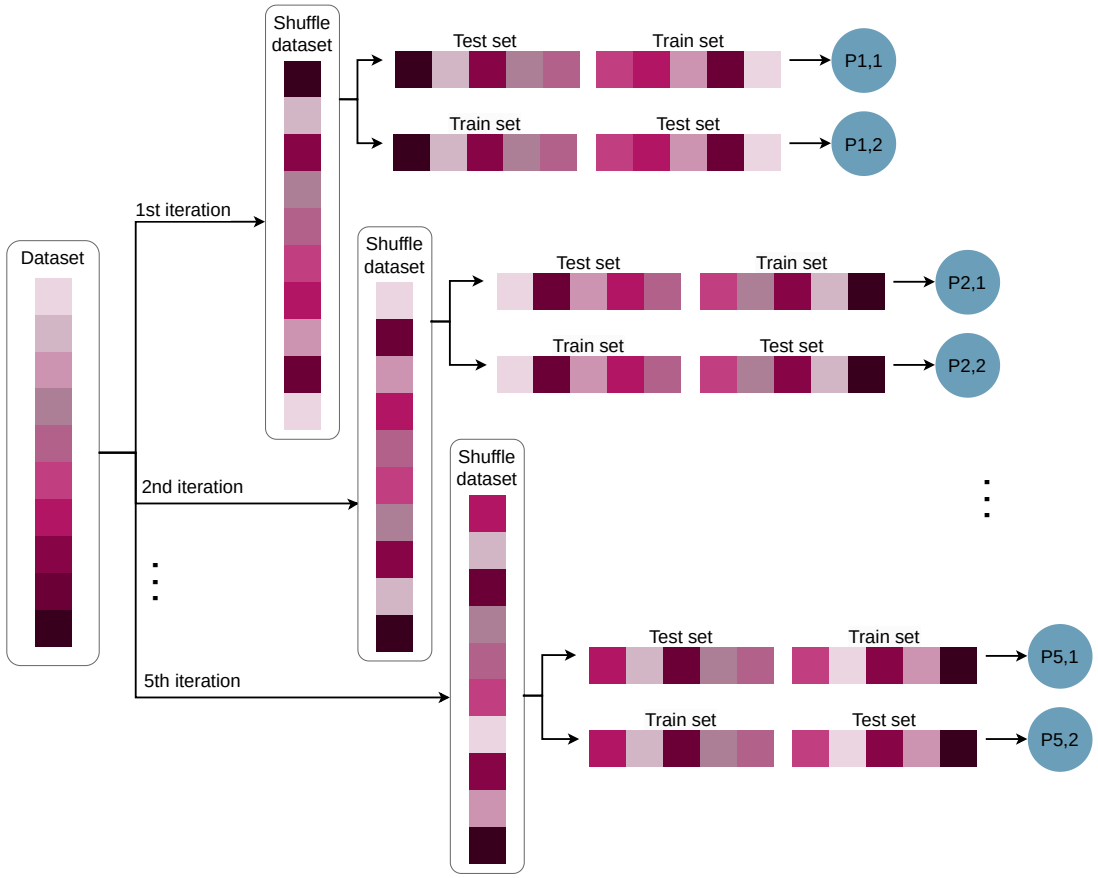


Figure 2.6: *5x2 cross validation*

Another way to determine *TS* and *VS* is bootstrapping, which is a suitable method, especially for small datasets. Samples may overlap more than in cross-validation. Bootstrapping draws examples with replacements from the learning set and creates a training and validation sets.

After training and validating the algorithms, they are compared in several ways. The misclassification error is checked using the performance metrics detailed at the beginning of this chapter, such as *Accuracy* or *Gmean*. The computational or time complexities of the methods and the cost of the classifier's decisions can be compared [153].

A crucial element is to conduct **statistical tests** that enable comparison of classifiers. An appropriate method should be selected depending on the number of compared algorithms and datasets. Tests are called parametric when the population distribution is strongly defined by certain assumptions, and nonparametric tests when these assumptions are weaker. In the case of two algorithms and one dataset, this may be a parametric *Two-Matched-Samples t test*. In case of multiple datasets, nonparametric *Sign Test* or *Paired t-test*. Comparing two algorithms regardless of how many datasets can be accomplished by the nonparametric *Wilcoxon's Signed-Rank Test for Matched Pairs*, which is based

on t test. When appropriately modified, the tests mentioned above can also compare many classifiers. However, referring to the definition of statistical tests, an *ANOVA* test (parametric) or a *Friedman's test* (nonparametric) are used when experiments involve the comparison of more than two methods on multiple datasets. The last two statistical tests should also use a post hoc test, such as the *Nemenyi* or *Bonferoni-Dunn Post hoc* test [72]. The *Bergmann-Hommel* or *Shaffer*'s procedure can be used in multiple comparisons [53].

Demsar in [37] argues that more preference should be given to nonparametric tests like *Wilcoxon* rather than parametric tests like *ANOVA*. Nonparametric tests are more likely to reject a null hypothesis. They do not assume normal distributions or homogeneity of variance. Tests can be performed for various performance measures and even computation time. Derrac et al. also agree with this statement about using nonparametric tests for evolutionary or swarm intelligence methods for continuous optimization problems in multi-problem examination because they are reliable [39].

The Wilcoxon signed-ranks test [37] compares two classifiers for each dataset by counting ranks for positive and negative differences. Hence, we have R^+ as the sum of ranks for all datasets on which the second algorithm won and R^- as the sum of ranks on which the first algorithm won. d_i is the difference in performances of two classifiers on the i-th dataset, with N of all datasets. If there are ties, i.e., $d_i = 0$, then R^+ and R^- receive equally half of the votes. When there is an odd number of ties, one rank is omitted.

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{2.18}$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{2.19}$$

where $rank()$ denotes a statistic function that tests the null hypothesis if the two sets of measurements come from the same distribution. For datasets $N > 25$, the statistic z is distributed normally

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \tag{2.20}$$

where T is the minimum of positive and negative sums, $T = min(R^+, R^-)$. If the significance of $p < 0.05$, the results obtained with the proposed method are statistically

significantly different from the reference method. However, for $p \geq 0.05$, the proposed method does not differ statistically significantly from the reference method.

2.3 Multi-objective optimization

Nature has always been an inspiration for researchers, including in the case of optimization algorithms. The idea of natural selection and Darwin's evolution gave rise to the *Genetic Algorithm* (*GA*). *GA* has a mathematical representation of evolution and uses concepts such as offspring population, crossover, and mutation. Observing the behavior of ants creating paths using pheromone concentration contributed to the Ant Colony Optimization (*ACO*) algorithm proposal. Artificial Bee Colony (*ABC*) is another algorithm based on animals, this time on bees, whose colony is divided into groups performing different tasks: employed bees (forager bees), onlooker bees (observer bees), and scouts. The phenomenon of fish schooling and bird flocking in nature also contributed to the proposal of the Particle Swarm Optimization (*PSO*) algorithm. Flower Pollination Algorithm (*FPA*) is a slightly newer proposition with a multi-objective version. These are a few examples of nature-inspired optimization algorithms [159].

The **optimization** process is finding the best solution from a pool of possible answers. Various algorithms solve the task and find its solutions. Exhaustive algorithms guarantee finding a solution if it exists and searching the entire space, but they are only suitable for small tasks. More advanced methods, such as *Genetic Algorithm*, should be used if the search space is ample.

The **Genetic Algorithm** is based on the principles of natural evolution and uses its definitions. A genotype (population) is a potential solution to a problem consisting of vectors. The algorithm searches the population in different directions until it finds the best solution. In each generation, solutions are evaluated by the objective (fitness) function, and only those that are good enough remain. They are then transformed by genetic operators such as crossover and mutation. Crossover involves combining parents' chromosomes and exchanging parts of them to obtain diverse offspring. A mutation randomly changes one or more genes in a chromosome, which provides variability in the population [102].

Another widespread algorithm is ***Differential Evolution*** (*DE*) algorithm [134]. *DE* works on the differences between individuals from the population. It is a single-criterion algorithm that combines the advantages of local search and genetic algorithms. It uses classical crossover operators and search mechanisms typical for topological spaces. *DE* searches the continuous space. Hence the solution obtained from this algorithm is real

numbers. *DE* is looking for new solutions close to the best in earlier iterations. However, it does not use a predefined scheme or distribution. Subsequent generations of DE lead to finding the optimal state. Fig. 2.7 presents this process. The algorithm selects three random points from the population X_{r_1}, X_{r_2}, and X_{r_3}, and then the difference between the two selected points $X_{r_2} - X_{r_3}$ is scaled by the λ control parameter $\lambda(X_{r_2} - X_{r_3})$. After adding this to the remaining point X_{r_1}, we obtain the current reference point and differential mutation vector $X_{r_1} + \lambda(X_{r_2} - X_{r_3})$ [47].

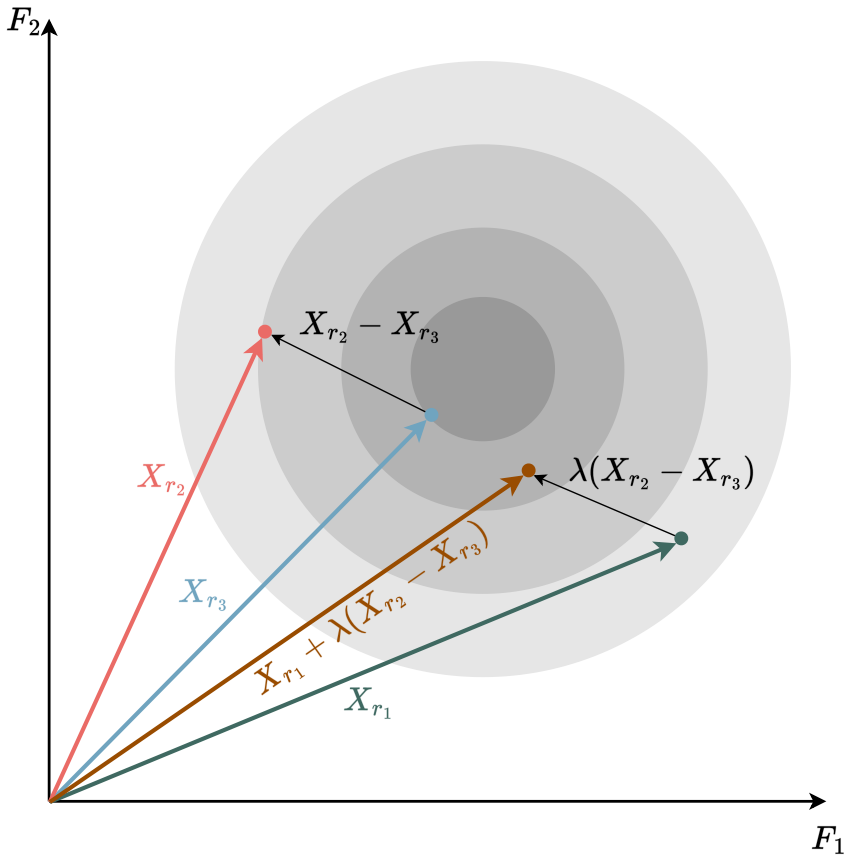


Figure 2.7: *Differential Evolution algorithm*

In real problems, one criterion is often insufficient, and many objectives are used. Single-objective algorithms can consider several criteria, but only in an aggregated form, reducing the function to returning one value. Simultaneous optimization of many contradictory criteria is called **multi-objective optimization** (*MOO*) or **multi-criteria optimization**. *MOO* is much more complex than single-objective optimization because the algorithm must find a trade-off between different criteria [42].

The criteria are successive objective functions $F_1, F_2, ..., F_R$, which can be minimized or maximized. For example, suppose the algorithm is designed to optimize only two criteria. In that case, we can graphically present it on a two-dimensional plot (Fig. 2.8). Each axis reflects one criterion, and each of them should be minimized. All found solutions are

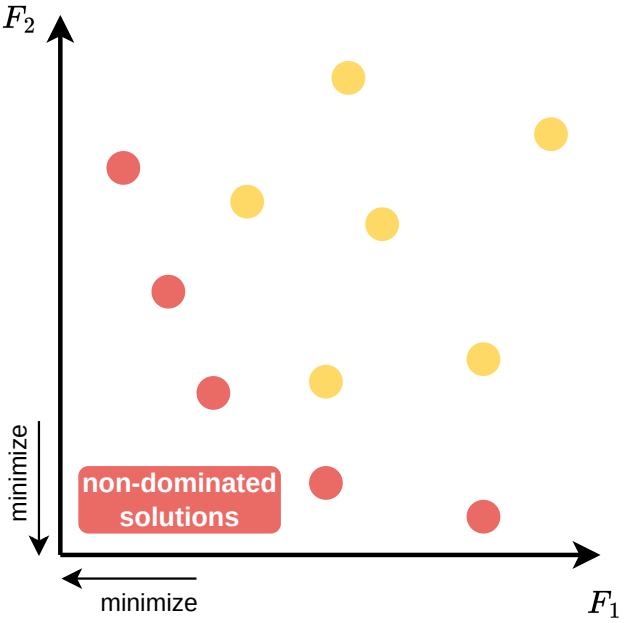


Figure 2.8: *Multi-objective optimization*

presented as two-colored dots because some of the solutions are non-dominated. These are solutions that dominate other solutions, i.e., the values of both criteria are better than for the dominated solutions. Non-dominated solutions are an approximation of a *Pareto Front* (*PF*). In Fig. 2.8, pink solutions dominate yellow ones. In a formal way, let F_i is an objective function, z_1 and z_2 are different solutions, and z_1 dominates z_2, then:

$$F_i(z_1) \leq F_i(z_2) \quad \forall_i \in 1, 2, ..., R \tag{2.21}$$

The answer to the problem using multi-objective optimization is a set of non-dominated solutions. The first issue is choosing the right solution. A set of solutions gives the user many options. The user has many criteria, so he may favor some of them and then may pay more attention to those solutions that achieve the best results for the most important criteria. Understanding the data is essential here, so the right person should make this decision. Choosing a solution is an advantage of multi-objective optimization because single-criteria algorithms return only one solution, and the user cannot directly influence the final choice. In some cases, this is a minus because the user is responsible for choosing the final solution, even though he still chooses from the non-dominated solutions, where there are no worse solutions.

Therefore, algorithms have been created to support this decision, belonging to the multi-criteria decision-making (*MCDM*), including *TOPSIS* [161], *VIKOR* [112], *PROMETHEE*

[17], *GAIA* [18], *SIR* [156], and *AHP* [122]. *PROMETHEE* is frequently used, so we also used it to select a solution from the non-dominated set in our several methods. *PROMETHEE* (*Preference Ranking Organization METHod for Enrichment of Evaluations*) has many versions using intervals, fuzzy numbers, or group decision-making. This method supports the user but does not make the final decision for him.

PROMETHEE applies weights to each criterion, determining each criterion's priority and making a decision based on the preference function. Preference degree means how much one alternative (solution) is preferred to another (z_1, z_2), and always has a value from 0 to 1.

$$0 \leq P_j(z_1, z_2) \leq 1 \tag{2.22}$$

If the criterion is maximized, the preference function can be defined as

$$\forall z_1, z_2 \in A \quad P_j(z_1, z_2) = F_j[d_j(z_1, z_2)] \tag{2.23}$$

where A is a finite set of possible alternatives and $d_j(z_1, z_2)$ is the difference of evaluations among two alternatives

$$d_j(z_1, z_2) = g_j(z_1) - g_j(z_2) \tag{2.24}$$

where g is a criterion.

The algorithm takes different parameters depending on the selected preference function. There are several types of functions, and from here, we have different types of generalized criteria: *Usual*, *U-shape*, *V-shape*, *Level*, *V-shape with indifference*, and *Gaussian*. The basis of the *PROMETHEE II* version is net flows and output in the form of a ranking of alternatives [113]. A frequently used combination is *PROMETHEE* with the visual interactive module *GAIA* [17].

Obtaining a set of solutions gives the user greater opportunities to find an excellent final solution, but the result is only sometimes satisfactory. Hence, the second problem is the quality of the resulting non-dominated set. Fortunately, various quality metrics from the literature allow us to extract more information from *PF*. Laszczyk and Myszkowski [84] presented a broad overview of 38 Quality Measures along with their advantages and disadvantages from various categories, such as *distance*, *cardinality*, *dominance*, and *volume*. Different measures answer different questions: how close is the approximation to the real *PF*, how many *PF* solutions are there, and how diversified are they.

The necessary element of *MOO* is algorithms specially adjusted to adopt more than one criterion. Most multi-objective optimization algorithms are global methods [32], where approximate solutions are returned. They may be grouped into: (i) *Weighted Objectives Methods*, (ii)*Hierarchical Optimization Methods*, (iii) *Trade-Off Methods*, (iv) *Methods of Distance Functions, Min-Max Methods*, (v) *Goal Programming Methods*, (vi) *Genetic Methods*.

Among the latter methods, *NSGA*, *NSGA-II*, *NPGA*, and *FFGA* [3] should be mentioned. One of their most popular is *NSGA-II* (*Non-dominated Sorting Genetic Algorithm II*) [35], a modification of single-criteria *GA*. First, *NSGA-II* initializes the population P_0, then sorts based on non-domination criteria and selects individuals based on rank and crowding distance. Then, it uses genetic operators and performs *selection*, *crossover*, and *mutation*, thus obtaining the offspring Q_0. Fig. 2.9 shows the t-th generation of the algorithm. The combined population, R_t of size $2N$, consists of the population P_t and offspring population Q_t. Then the algorithm performs non-dominated sorting of solutions, and these solutions go to non-dominated sets A_1, A_2, A_3, ..., which means that front A_1 dominates A_2, A_2 dominates A_3, etc. The set A_1 contains the best solutions with R_t. Only the best solutions that fit up to size N (dashed line on the graph) are transferred to the next population P_{t+1}, i.e., crowding distance sorting is performed. The advantages of *NSGA-II* are speed in non-dominated sorting and crowded distance estimation [162]. Lin et al. [92] present *NSGA-II* modification and test various method variants.

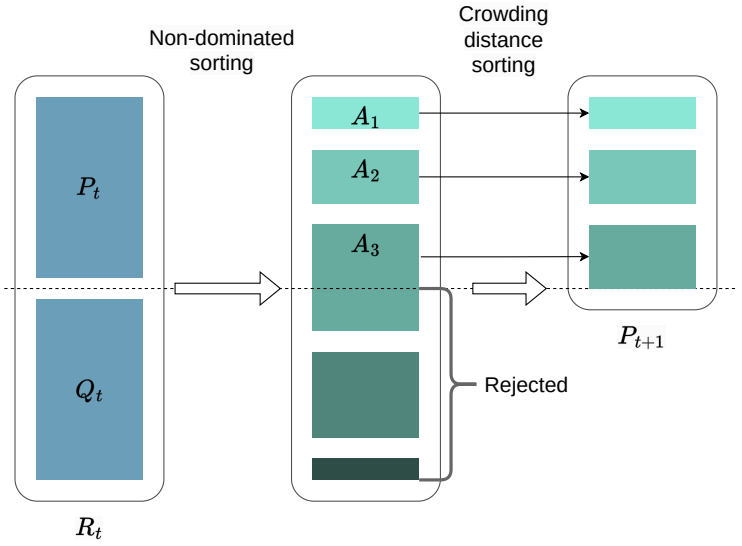


Figure 2.9: *NSGA-II algorithm*

Another proposition on how to solve optimization tasks is worth mentioning. One uses

the surrogate-assisted *Particle Swarm Optimization* with the Pareto active learning algorithm [96], which has a relatively low computational cost, fast convergence, and good diversity. Convergence is good when the solutions are close to the Pareto front, and good diversity in the objective space means that the solutions are evenly distributed in space and are not just at one point.

Multi-objective evolutionary algorithms (*MOEA*s) are eagerly developed and bring significant benefits, especially in large-scale *MOO* problems. In this case, the decision space is grouped into several subspaces, is reduced, or novel search strategies are used [139]. The algorithm based on decomposition (*MOEA/D*) is gaining popularity. It decomposes an optimization problem into several scalar optimization subproblems and optimizes them simultaneously [165]. Nguyen et al. [108] modified the *MOEA/D* algorithm and proposed two decomposition methods based on multiple reference points, static and dynamic. The developed method gave satisfactory results in the feature selection in the classification. Ma et al. [99] proposed the *LSMOEA/D* method using reference vectors in the control variable analysis. The experiments confirmed high quality for the vast test problems with 2–10 objectives and 200–1000 variables.

In multiple newer works, the *MOEA/D* method is the most potent competitive method, and many proposed multi-objective algorithms are based on it, compared to the *NSGA-II* method, which usually achieves weaker results and is treated as a reference point [120, 140, 166]. In [25], *NSGA-II* is even omitted, and the proposed method uses support vector regression and *MOEA/D*.

2.4 Multi-objective optimization in ensemble learning

Considering the advantages of *MOO*, this optimization is widely used for classification and is suitable for various tasks. *MOO* performs parameter tuning and selects a set of appropriate ones for classifiers, selects features, creates ensemble classifiers, and is used to classify difficult data, e.g., imbalanced.

Feature selection in classification meets two contradictory criteria. It strives to obtain the highest possible model performance while minimizing the number of features. Therefore, it is a classic example of a *MOO* where various criteria must be compromised. *MOO* methods for feature selection result in non-dominated feature subsets and a more interpretable model than deep learning. Jiao et al. [74] presented an extensive survey and proposed dividing multi-objective feature selection methods into six categories depending on their components. These categories along with examples are: *Solution Representation*

[105], *Evaluation Function* [168], *Initialization* [155], *Offspring Generation* [95], *Environmental Selection* [146], and *Decision Making* [13].

Proposing a new framework in [90] led to a more diverse set of features of the non-dominated solutions in less time. The *MOO* task is solved here by an auxiliary single-objective feature selection task, which creates single subtasks from a multi-objective problem and searches the solution space in distinct directions. The method presented in [75] reformulates the feature selection task into a constrained form and assumes performance constraints for each solution in the form of a feature subset.

Several papers combine *SVM* classifiers with optimization algorithms that solve the feature selection problem. The article [164] used *NSGA-III* for the feature subset selection and *CNN-SVM* (*Convolutional Neural Network - Support Vector Machine*) for software defect prediction with an imbalance problem.

Researchers dealing with neural networks also often employ *MOO*. Jin et al. [160] solved the problem of multi-objective optimization of the structure and parameters of the neural network using evolutionary methods (*NSGA-II*). The *MO-SELM* method [154] has been tested for classification and regression. It relies on optimizing parameters and structure learning of the *Extreme Learning Machine* network to cope with the overfitting problem. *GEMONN* [158] employed a gradient-guided evolutionary approach containing the advantages of gradient descent and evolutionary algorithms to train deep neural networks. Optimization is used to determine weights for the network, while multi-objective optimization is used simultaneously for the network sparsity and the training loss. *MOO* is often applied to real-world problems, e.g., in the iron and steel industry. In [148], a multi-objective convolutional neural networks ensemble learning method for quality prediction was proposed.

Mierswa [103] showed the possibility of using multi-objective optimization techniques in *SVM* learning, pointing out that thanks to this approach, it is possible to turn away from aggregated optimization criteria as a combination of opposing criteria. Additionally, Pareto-optimal solutions allow complexity analysis so the user can easily see which solutions are overfitted. The combination of *SVM* classifier to detect malicious traffic and a *Genetic Algorithm* to optimize hyperparameters is used in the article [123].

Chandra and Yao proposed *DIVACE* (*DIVerse and Accurate Ensemble Learning Algorithm*) [26], which employs multi-objective optimization to the ensemble learning task to find a trade-off between diversity and accuracy. Abbas [1] developed *Memetic Pareto Artificial Neural Network* (*MPANN*), which optimizes similar criteria that are essential when creating ensembles. Gu et al. [61] focused on classifier ensemble generation,

proposing a solution that is also a compromise between accuracy and diversity ensemble. They also showed that the proposed solution could outperform single-objective methods.

In [150], the authors optimize the weights of the models in the ensemble and select the non-dominated solution using the *PROMETHEE* method. Other works focus on ensembles of *DT* [76], *recursive networks* [129], or *fuzzy rules* [70], which prove the possibility of using this approach. However, the number of works devoted to employing multi-objective optimization to classifier ensemble design is relatively small, especially compared to the common problem of designing the ensemble using a single criterion. Answers to critical questions require further research, i.e., which models are best suited to the multi-objective task and whether it is possible to develop algorithms that effectively combine different decision models. Developing methods for forming the base classifier and creating combination rules that connect them in the multi-objective optimization task is also necessary.

Fletcher et al. [51] proposed a non-specific ensemble classification algorithm that uses multi-objective optimization to select base classifiers and minimize user-defined parameters. Ribero and Reynoso-Meza [118] developed a two-stage ensemble learning framework that generates a set of diverse classifiers and prunes a pool of models. The same authors [6] analyze different *MOOD* approaches for ensemble learning and propose a taxonomy of multi-objective ensemble learning. Olivera et al. [110] employed multi-objective optimization to select the base classifier's valuable features and then choose the best ensemble line-up. Onan et al. [111] developed an ensemble method that employs a static classifier selection involving majority voting error, forward search, and a multi-objective differential evolution algorithm. Liang et al. [89] described an ensemble learning model based on multimodal multi-objective optimization. Asadi and Roshan [9] formulated an intriguing proposition that focuses on the bagging procedure, considering the number of bags and the diversity simultaneously of the trained classifiers.

Only a few works on imbalanced data classification employ multi-objective optimization methods. It is worth mentioning the work of Bhowan et al. [11], who proposed to build a classifier ensemble based on Pareto-optimal classifiers. In turn, Soda [130] suggested training the classification system named *Reliability-based Balancing*, using multi-objective optimization methods and maximizing two criteria related to the frequency of correct decisions and *Gmean*. They simultaneously used the data preprocessing technique based on feature and prototype selection obtained from multi-objective optimization.

Li et al. [88] proposed a data preprocessing method *Adaptive Multi-objective Swarm Crossover Optimization*, which used both over- and under-sampling simultaneously. This approach selected the best proportion between majority and minority samples by multi-objective optimization. Bhowan et al. [12] proposed a two-step approach to evolving

ensembles using genetic programming for imbalanced data classification. The optimal classifiers non-dominated set form an initial pool of classifiers, and then ensemble pruning methods based on genetic programming were employed. Fernandez et al. [48] employed a decision tree ensemble and multi-objective optimization to find the best combination between feature and instance selections for the multi-class imbalanced task.

Felicioni et al. [46] developed an algorithm that took fourth place in the *ACM RecSys Challenge 2020*, organized by *Twitter*. The challenge aimed to predict the probability of user engagement based on past interactions on the *Twitter* platform. Authors employ feature extraction and gradient boosting for decision trees and neural networks and build the ensemble using multi-objective optimization. A broad review of evolutionary computation methods for classifying imbalanced data is presented in [115]. A particular case of difficult data is an imbalanced data stream, for which *MOO* algorithms are also helpful. It was confirmed in [143], where a Pareto-based ensemble for imbalanced and drifting data streams was proposed.

Chapter 3

Feature selection method

This chapter aims to propose a feature selection algorithm based on multi-objective optimization. Its quality will be compared with the classical approach and the genetic algorithm in single- and multi-objective versions. Additionally, the computational complexity of the proposed method will be estimated. Extensive research for each dataset permits us to answer research questions about the classifier's choice impact on the proposed method, the differences between the aggregated criterion and the multi-objective approach, and the comparison of the proposed method with state-of-the-art methods.

Feature acquisition involves a cost assigned to each feature, and not all feature selection methods consider this cost. When the upper limit of the budget is predetermined, high-cost and non-informative features should not be selected. It is an essential aspect of cost-sensitive learning that is worth considering when designing feature selection methods. We propose a feature selection approach and assume that obtaining a lower total test *cost* and comparable *Accuracy* (not worse) to reference methods is possible.

Let us propose two objective functions that could be used for feature selection. Firstly, the maximum *Accuracy* score indicated as *GA-a*.

$$\text{maximize } GA\text{-}a = Accuracy \tag{3.1}$$

Secondly, we may aggregate the *Accuracy* score and acquisition *cost* of used features to obtain a cost-sensitive classifier marked as *GA-ac*.

$$\text{maximize } GA\text{-}ac = \frac{Accuracy}{cost} \tag{3.2}$$

This criterion does not provide a precise solution to the individual *Accuracy* and *cost* values because the function value is aggregated. Using only one-objective can be insufficient, and even the aggregating process is not enough. Therefore, from the point of view of a user who wants to know the exact values of both metrics, a better solution is to use multi-objective optimization, where each criterion is considered separately [44]. From several algorithms, we chose *NSGA-II* (Non-dominated Sorting Genetic Algorithm II) [35], the updated multi-objective version of *GA*, for the feature selection problem [128]. In the experiment, *NSGA-II* is applied with two fitness functions. The *Accuracy* has to be maximized and the total *cost* – minimized.

$$\begin{cases} \text{maximize } F_1 = Accuracy \\ \text{minimize } F_2 = cost \end{cases} \tag{3.3}$$

The diagram of the general genetic algorithm is in Fig. 3.1. The binary representation is an example of six features of the *liver* dataset. The bit string is a vector of features called an individual or a solution, where 1 means that the algorithm selects the feature and 0 – is not selected. In the beginning, random sampling is performed, so the initial set of solutions is created. Then, the *binary tournament random mating selection* is used. N-individuals are selected in each tournament, where $n = 2$ in our case. Individuals are compared, and the winner is taken to the next generation population. It is a simple and efficient solution that ensures diversity [116]. Next, two genetic operators are applied to produce new offspring: the *binary point crossover* and the *bitflip mutation*. The selection is used to choose significant solutions to create the population, and genetic operators explore the search space. As shown in Fig. 3.1, the crossover swaps the part of the bit string, and the mutation replaces the bit with the opposite value. The search is over when the algorithm reaches the population size (steps 1-5 in Alg. 3.1).

The *NSGA-II* algorithm (described in Sect. 2.3) returns the non-dominated set of solutions (called the non-dominated solution set PS) from which one *solution* must be chosen to contain a subset of the best features (step 6). These features are used to learn the classifier and obtain the best performance and the lowest total *cost*. However, the best features are different when there are distinct user's expectations. Sometimes, the total test *cost* or the *Accuracy* is more important, but sometimes, there is a need to have two good criteria. Unfortunately, during the experiments, we did not have access to a user who could specify the preference. Hence, multi-criteria decision rules were proposed to select a solution from an estimated Pareto front. It is also done to confirm the first part of the research hypothesis of this work that the proposed methods based on multi-objective optimization allow obtaining solutions that are no worse than those based on single criteria. Therefore, we applied three approaches to select solutions:

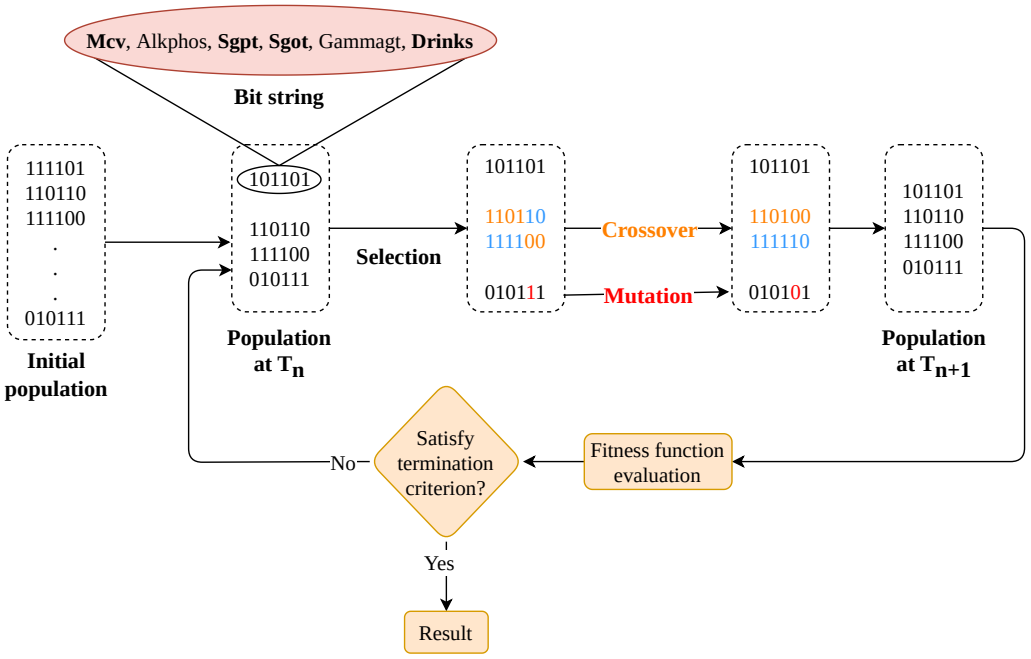


Figure 3.1: *Diagram of the genetic algorithm and operators*

1. *NSGA-a* – the criterion based on the maximum *Accuracy*.
2. *NSGA-c* – the solution with the minimum *cost*.
3. *NSGA-p* – the *PROMETHEE II* approach [77] described in Sect. 2.3.

At the end, we obtain a classifier model (step 7), different depending on the choice of classifier in the experiments. In the tested case, it will be *CART Decision Tree Classifier*, *Support Vector Machines*, *Gaussian Naive Bayes* or *K-Nearest Neighbors Classifier*.

Computational complexity analysis

Several methods have been proposed above, which consist of different components. Therefore, the computational complexity is given for individual components. The single-criteria genetic algorithm has a computational complexity of $O(\frac{dim}{2})$, where dim is the size of the search space [121]. For the multi-objective case, the *NSGA-II* algorithm is used. Its complexity is $O(MU^2)$, M is the number of objectives, and U is the population size [35]. The research used four different classifiers, so their complexity should also be considered. The complexity of the *CART Decision Tree* is $O(dN \times log_2 N)$, where d is the number of attributes, and N is the number of samples [124]. The time complexity of *Support Vector Machines* is $O(N^3)$ [2]. *Gaussian Naive Bayes* and *K-Nearest Neighbours* classifiers have the same complexity, $O(N \times dim)$ [38].

Algorithm 3.1: *Classification with multi-objective feature selection*

Input:

$\mathcal{LS} = \{(x_1, i_1), (x_2, i_2), \ldots, (x_N, i_N)\}$ – learning set
$Validate()$ – validation function
$Evaluate()$ – evaluation function
$PROMETHEE\ II$ – algorithm to choose one solution

Symbols:

U – population size
x – feature vector
Ψ – classification algorithm
PS – non-dominated solution set

Output:

m – final model classifier

1: **for** $i = 1$ to U **do**
2: $x_i \leftarrow$ features from $\mathcal{LS}$ ▷ after using genetic operators (selection, crossover, mutation)
3: $Validate(x_i)$
4: $PS \leftarrow Evaluate(x_i)$ ▷ based on eq. 3.3
5: **end for**
6: $solution \leftarrow PS$ ▷ according to the best *Accuracy*, the least *cost* or *PROMETHEE II* function
7: $m \leftarrow \Psi(solution)$

3.1 Experimental evaluation

We compare all methods described previously and believe the multi-objective optimization approach can achieve a more inexpensive total *cost* without an *Accuracy* drop compared to other methods. To prove our hypothesis, we conducted an experimental evaluation. Our research tries to find answers to two research questions:

RQ1: **How do the proposed feature selection methods work for a given classifier?**

RQ2: **How does multi-objective optimization differ from an aggregated optimization criterion?**

RQ3: **Can feature selection methods based on multi-objective optimization outperform classical feature selection and methods based on one-objective optimization?**

3.1.1 Setup

Datasets with the corresponding features' cost are obtained from the *UCI Machine Learning Repository* [91]. All of them are medical datasets where the total cost of tests is essential, so classification is a challenging task. The information about the number of examples, attributes, and classes is in Table 3.1. The aim of the first dataset, *heart-disease*, is to predict if a patient has heart disease. The *hepatitis* dataset contains information about patients with Hepatitis disease and decision 1 or 2 (die or survive). The *liver-disorders* dataset has the smallest number of features. Based on them, the decision of a person who has alcoholism is made. The *pima-indians-diabetes* contains only female medical data from the Pima Indians group (Native Americans), in which the class says if a person has diabetes or not. The last dataset, *thyroid-disease*, is the biggest and has many features. It contains three classes that decide if an individual is average or suffers from hyperthyroidism or hypothyroidism (1, 2, or 3). Our experiment only considers these datasets where information about feature acquisition cost was given.

Table 3.1: *Datasets*

DATASET	EX.	ATTR.	CLASSES
heart-disease	303	13	4
hepatitis	155	19	2
liver-disorders	345	6	2
pima-indians-diabetes	768	8	2
thyroid-disease	7200	21	3

The project is implemented in the *Python* programming language, and it is available in the *GitHub repository* [1] along with results from the experiment. A few libraries were used: *Pymoo* [14], *Matplotlib* [67], *Pandas* [149], *Numpy* [109] and *scikit-learn* [114]. From the last one, we used the following classifiers with the default parameters:

- *DT – CART Decision Tree Classifier* (the criterion is Gini impurity, the splitting strategy is *best*, the minimum number of samples required to split an internal node equals 2)
- *SVM – Support Vector Machines* (the parameter $C = 1$, the kernel is RBF, γ is scaled)
- *GNB – Gaussian Naive Bayes* ($variance = 1 \times 10^{-9}$)

[1] `https://github.com/joannagrzyb/moofs`

- *kNN – K-Nearest Neighbors Classifier* (the number of neighbors is 5, weights are uniform, the algorithm used to compute the nearest neighbors is automated, leaf size is 30, the metric used for distance computation is Minkowski, power parameter is 2)

Before experiments, all datasets must be preprocessed:

1. Missing values were replaced with the most frequent ones.
2. Data and features' costs were normalized.

The number of examples is relatively tiny, so 5×2 *CV* was used to avoid overfitting. Mechanisms of feature selection described in the beginning were used. Genetic algorithms *GA* and *NSGA-II* have the same parameters: the population size is 100, and the evaluation's number is 1000. These algorithms use *Random Sampling*, *Two Point Crossover* (*GA* uses *Half Uniform Crossover*), and *Bitflip Mutation*. A constraint was imposed as a percentage of selected features to prevent the algorithms from selecting all features. The exact value was used for the *K-best* algorithm.

The proposed algorithm finds many solutions and returns only the non-dominated set while using multi-objective optimization. Fig. 3.2 shows three non-dominated solutions in black dots, each with corresponding values of the *Accuracy* score and the total *cost* of the selected features. The greater the *Accuracy*, the higher the *cost*, so choosing an appropriate solution is crucial. Hence, we applied three criteria described in the beginning to select the best solution. As it was mentioned earlier, selecting only one solution from the set obtained from the multi-criteria optimization algorithm was necessary to compare it with the single-objective approach. Therefore, multi-criteria decision making techniques were used. If this were a real problem, the user could decide the most suitable solution. Because there is no end user in the cases studied, the solution selection must be done more automatically. One of the techniques is *PROMETHEE II*. In research, *PROMETHEE II* approach takes a weight of 0.4 for the *Accuracy* and a weight of 0.6 for the *cost*.

Feature selection methods using optimization algorithms were compared with the classical approach, and we refer to it as *FS* (Feature Selection). The *Chi-square* test statistic is applied to the *Select K-best* function. The experiment compares six feature selection methods and four classifiers tested on five datasets. The number of features changes from 1 to the maximum features in the dataset. All approaches to the feature selection problem with the abbreviation coming up in figures are presented in Table 3.2.

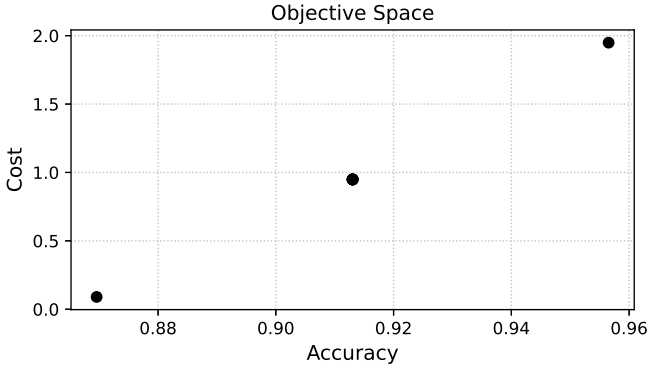


Figure 3.2: *Non-dominated solutions, dataset: hepatitis*

METHODS	OBJECTIVES	CRITERIA	ABBR.
Select K-best (Chi-square) Feature Selection	-	-	*FS*
Genetic Algorithm	max. *Accuracy*	-	*GA-a*
	max. (*Accuracy*/*cost*)	-	*GA-ac*
Non-dominated Sorting Genetic Algorithm II	max. *Accuracy* min. *cost*	max. *Accuracy*	*NSGA-a*
		min. *cost*	*NSGA-c*
		PROMETHEE	*NSGA-p*

Table 3.2: *Methods' abbreviation*

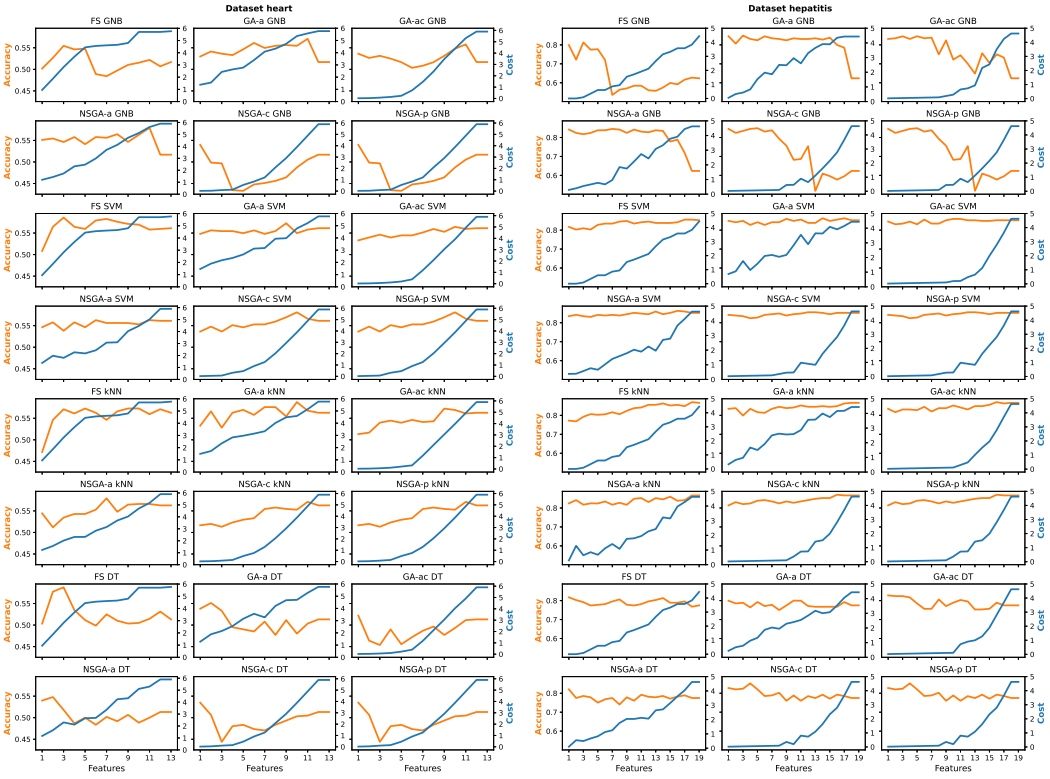
3.2 Results

Firstly, the preprocessing experiments were run, and then, based on the *Accuracy* score measuring the performance, methods were compared. Twenty-four micro charts show results for each dataset, in which an orange line represents the *Accuracy* and a blue line – the total *cost*. The *cost* of each feature is normalized to a value from 0 to 1, so the total *cost* is the sum of these values, not the original ones from the *UCI*, and it is on the second y-axis on the right. The y-axis on the left contains the *Accuracy* score, and the x-axis – the number of features.

Fig. 3.3a shows the results for the *heart* dataset. For optimization methods, the total *cost* is similar to the exponential function. Unlike the classical approach *FS*, the total *cost* grows very fast. For *SVM* and *kNN*, the *Accuracy* is stable for all methods, so choosing the smaller number of features is cost-effective because the *Accuracy* is almost the same, but the total *cost* is much smaller for *GA-ac*, *NSGA-c*, and *NSGA-p*. In this

case, the optimal number of features is 4 or 5. The tendency is different for remaining classifiers, and too many features lead to a deterioration of the classification's quality.

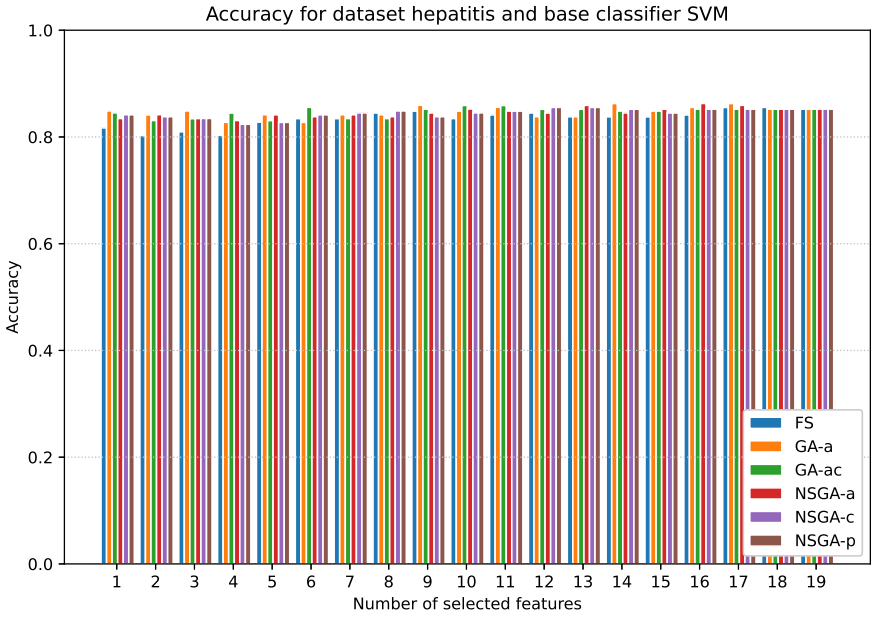
Fig. 3.3b shows the results for the *hepatitis* dataset. Using all features to learn the classifier is not always the best idea. This dataset contains 19 features, and for *GNB*, many features disturb good classification. We can obtain the same *Accuracy* level for other classifiers but a much smaller total *cost* using only five features and optimization methods with the *cost* criterion. Fig. 3.4 shows the *Accuracy* and total *cost* values for all tested approaches using the *SVM* classifier in the *hepatitis*. The *Accuracy* is very stable among all methods and through various selected features. Furthermore, as we observed earlier, the total *cost* is much smaller for the *GA-ac*, *NSGA-c*, and *NSGA-p* methods.



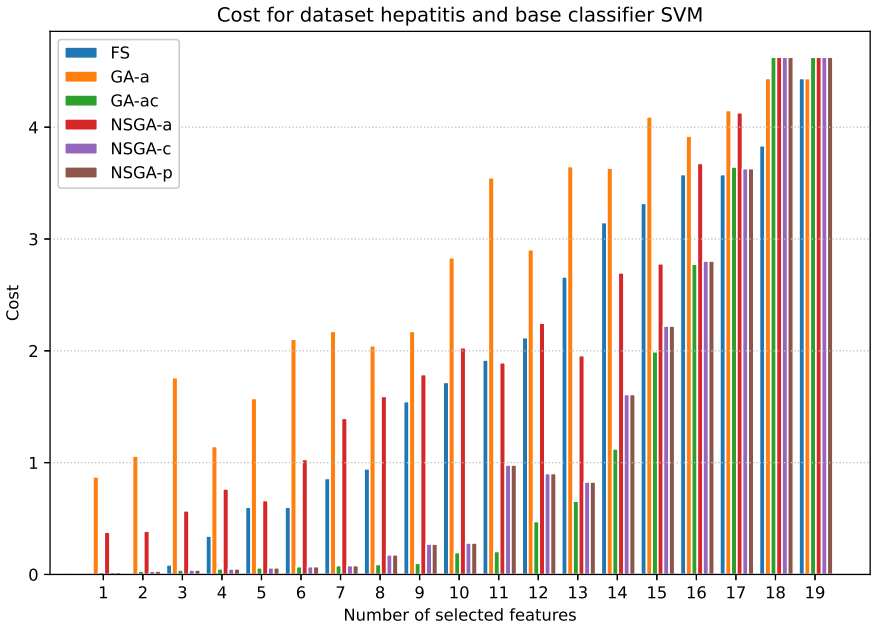
(a) *heart dataset* ***(b)*** *hepatitis dataset*

Figure 3.3: *Accuracy and cost results for all methods depend on the number of features*

The *liver* dataset (Fig. 3.5a) has the smallest number of features, so the difference between classical approaches and optimization ones is slight, especially for the *cost*, which is almost linear. However, the *Accuracy* for non-classical methods has a bigger value than for *FS*. The optimal feature number is 3 for *SVM* and *kNN*, where methods prefer the *Accuracy* (*GA-a*, *GA-ac*, *NSGA-a*) and keep the low total *cost*. Even if there is no need to use the cost-sensitive classifier, it is better not to use *FS* because it has a more minor performance.



(a) Accuracy

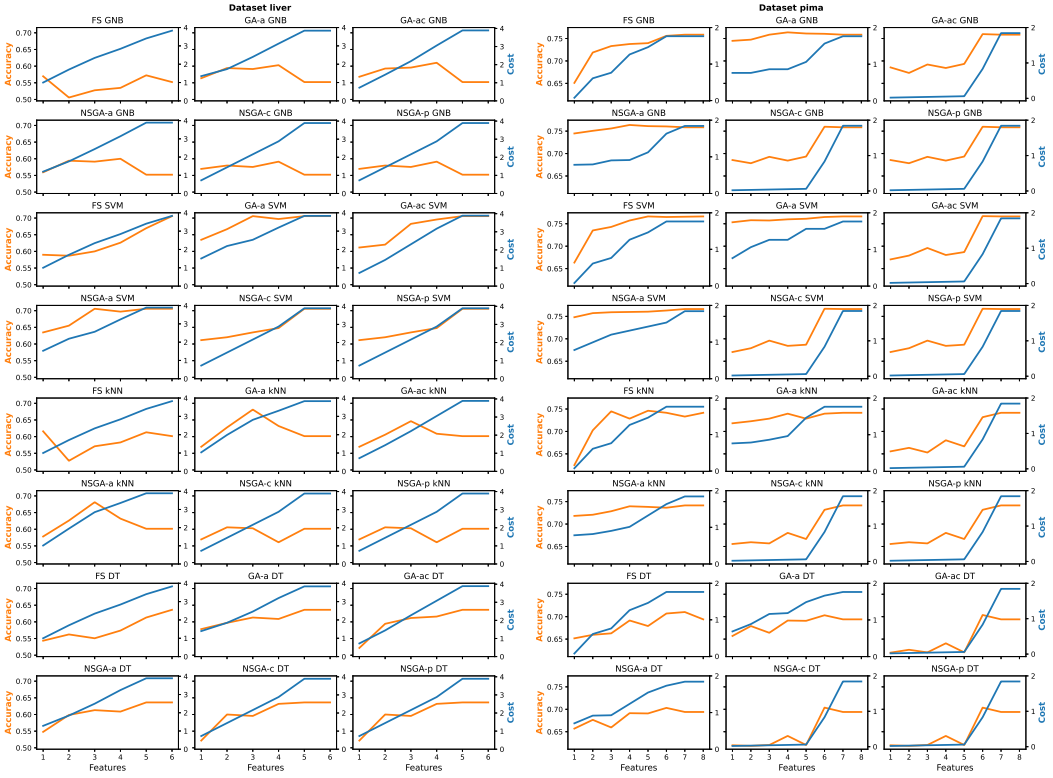


(b) cost

Figure 3.4: *Bar charts for dataset hepatitis and SVM classifier*

In the *pima* dataset in Fig. 3.5b, the *cost* is the smallest for methods with the *cost* criterion (*GA-ac*, *NSGA-c*, *NSGA-p*), but they obtain minor *Accuracy* simultaneously. As in the previous case, the optimal number of features is 3. At that point, the *GA-a* and *NSGA-a* for *GNB* achieve the highest *Accuracy*, over 75%, and the lowest *cost* under 1.

As in the previous dataset, the *thyroid* dataset (Fig. 3.6) has a similar *cost* shape to the



(a) *liver dataset* ***(b)*** *pima dataset*

Figure 3.5: *Accuracy and cost results for all methods depend on the number of features*

GA-ac, *NSGA-c*, and *NSGA-p* methods, and they achieve minimal total *cost*. For them, along with *SVM* and *kNN*, the optimal number of selected features is 9, with a *cost* close to 0 and an *Accuracy* of around 95%. Overall, for this dataset, the classification quality is much bigger than in other datasets because the *thyroid* has a few thousand times more instances.

3.3 Lessons learned

After conducting experiments, we can answer the research question posed at the beginning of this section:

RQ1: **How do the proposed feature selection methods work for a given classifier?**

GNB is not recommended as the *Accuracy* is usually smaller than for other classifiers, and it has a more significant discrepancy among different numbers of features. Otherwise, *SVM* and *kNN* give the stable *Accuracy* score among all feature selection techniques in *heart-disease*, *hepatitis*, and *thyroid* datasets. Unlike these data,

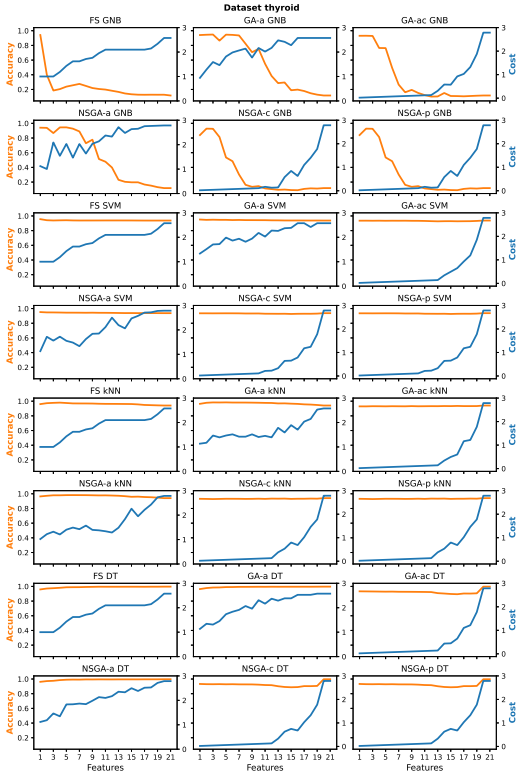


Figure 3.6: *Accuracy and cost results for all methods depend on the number of features for thyroid dataset*

the *Accuracy* varies in each technique through different numbers of selected features from 5% to 10% in *liver-disorders* and *pima-indians-diabetes* datasets. *DT* is similar to other methods.

RQ2: **How does multi-objective optimization differ from an aggregated optimization criterion?**

Suppose there is a need for a good performance classification and a low total *cost*. In that case, it is worth considering the multi-objective optimization algorithms to select the best features. The quality of the classification depends on the data. However, it can be helpful in the medical environment, where some tests can be costly. Thanks to this approach, a person can select only tests that give good results during the classification of the disease, and at the same time, the *cost* will be low.

RQ3: **Can feature selection methods based on multi-objective optimization outperform classical feature selection and methods based on one-objective optimization?**

The tested methods use a genetic algorithm to search for appropriate features to achieve a level of quality comparable to the classical feature selection approach. An important factor is the selection of appropriate criteria; when selecting a larger

number of them, multi-objective methods give an advantage in better adaptation to the problem. The advantage of multi-criteria optimization is the obtained set of solutions from which the user can choose the solution that best meets his/her needs.

The chapter aimed to propose a method and compare a few different feature selection approaches in the cost-sensitive classification task. The selection of *K-best* features with the *Chi-square* statistic, *Genetic Algorithm* in the form of the one-objective and multi-objective optimization were applied to classifiers, such as *Gaussian Naive Bayes*, *Support Vector Machines*, *K-Nearest Neighbors*, and *DT* classifier. The experiments showed that the proposed approach achieves accuracy similar to other methods but simultaneously considers the cost of features and strives to minimize it. Depending on the dataset, the one-objective or two-objective methods should be used to obtain better results than the classical approach, so our proposition is a promising approach. The work was published at the conference [56].

Chapter 4

SVM Ensemble with Multi-Objective Optimization Selection

This chapter proposes the *SEMOOS* algorithm that optimizes criteria related to the prediction quality of minority and majority classes. *SEMOOS* returns a pool of non-dominated solutions from which the user may choose the model that best suits him. The proposed approach trains an *SVM* classifier ensemble for the imbalanced data classification task. The experimental evaluations on many benchmark datasets confirm the proposed method's usefulness and answer research questions related to the proposed method's parameters, classification quality compared to other variants of the proposed method and reference methods, and diversity.

Among imbalanced data classification methods, one of the most promising directions is using models based on classifier ensembles. Two elements are essential in ensemble learning. On the one hand, the algorithm should obtain the best possible prediction quality, and on the other hand, it must ensure appropriate diversification of the base classifiers. Most of the work on classifying imbalanced data analyzes simple metrics such as *Recall*, *Precision* or *Specificity*. However, aggregate metrics such as *Gmean*, *AUC*, or *F-measure* are adopted due to the desire to express the quality assessment of a method by a single value. Their undoubted disadvantage is that they assume a fixed relationship between simple criteria, e.g., in the case of *Gmean*, it is the geometric mean of *Precision* and *Recall*. It is also worth noting that these criteria ignore the imbalance ratio or the misclassification costs on objects of different classes. Additionally, the values of these aggregated criteria are ambiguous. Based on their values, it is unclear how the given model behaves, i.e., what values the simple criteria take since a given value of

the aggregated criterion can be achieved for many pairs of *Precision* and *Recall* values. Considering the above, it seems interesting to consider the problem of classifier learning for imbalanced data as a multi-criteria optimization task. As a result of such algorithms, we should obtain not a single solution but a set of non-dominated solutions, from which the selection of a single solution can either be automated or left to the user to decide.

Let us propose the *SVM Ensemble with Multi-Objective Optimization Selection – SEMOOS* method dedicated to training an *SVM* classifier ensemble for imbalanced data [59]. Its pseudocode is presented in Alg. 4.1 and in Fig. 4.1. The main idea is to find a pool of *SVM*s that gives a diversified ensemble. To achieve that, we look for the setting of two parameters C and γ of the *SVM RBF* kernel (Sec. 2.2). Additionally, the feature selection for each base classifier is performed to ensure the high diversity of the ensemble. The multi-objective optimization is used to select the best *SVM* parameter setting, including feature selection [136]. When *Pruning* is not used, the entire set of solutions obtained from multi-objective optimization is used to build the classifier ensemble.

Bootstrapping generates data subspaces. Optimization based on this data results in differentiated *SVM* parameters from a given range and determines which features will be selected. The best non-dominated solutions were used to create a model, and then such a model was added to the ensemble of classifiers.

4.1 Algorithm

Let us quickly analyze Alg. 4.1, which starts with the learning set $\mathcal{LS}$ as an input. Suppose the center-based *Bootstrapping* mechanism is enabled ($Bootstrapping = True$). In that case, it is repeated W-times and divides each dataset fold into subsets S_i. A root r is selected randomly from the set of examples only for the first iteration (r_1), and this point becomes a center $\overline{x}$. The distribution list D_i is built based on the sum of distances D and every distance from the root r to each sample in the $\mathcal{LS}$. Then, D_i is used as a probability in the sampling with replacement. It creates the subset S_i composed of examples from $\mathcal{LS}$. The following root r_{i+1} is the furthest point from the center of mass. If x_c is the center of the time $k-$examples and d is the dimension of the x, then the center of $k+1$ examples is given by the following formula:

Algorithm 4.1: *SEMOOS, SEMOOSb, SEMOOSbp*

Input:

$\mathcal{LS} = \{(x_1, i_1), (x_2, i_2), \ldots, (x_N, i_N)\}$ – learning set

$SVM()$ – SVM classifier training method based on the SVM parameters C, γ, and selected subset of features $\hat{x}$

Symbols:

Bootstrapping – boolean parameter *Bootstrapping*

Pruning – boolean parameter *Pruning*

W – number of iterations of center-based bootstrapping

Output:

Π – pool of the SVM base classifiers

$\Pi \leftarrow \emptyset$
if *Bootstrapping* **then**
  for $i = 1$ to W **do**
    $solutions \leftarrow \emptyset$
    if $i == 1$ **then**
      $\overline{x} \leftarrow r_1$
    end if
    $D \leftarrow \sum_{k=1}^{N} dis(r_i, x_k)$
    for $k = 1$ to N **do**
      $D_i(k) = \frac{D - dis(r_i, x_k)}{(N-1)D}$
    end for
    $S_i \leftarrow$ sampling with replacement from $\mathcal{LS}$ according to distribution list of D_i
    choose r_{i+1} that $dis(r_{i+1}, \overline{x}) = \max_{k \in 1,N} dis(x_k, \overline{x})$
    $\overline{x} \leftarrow center(\overline{x}, i, r_{i+1})$ according to eq. 4.1
    $PS \leftarrow$ Optimization(S_i)
    for each $(C, \gamma, \hat{x}) \in PS$ **do**
      if *Pruning* **then**
        calculate *Precision* and *Recall* for $(C, \gamma, \hat{x})$
        unique $\leftarrow$ find records in *solutions* with the same *Precision* and *Recall*
        if *unique* $== \emptyset$ **then**
          $solutions \leftarrow solutions \cup (C, \gamma, \hat{x})$
        end if
      else
        $solutions \leftarrow solutions \cup (C, \gamma, \hat{x})$
      end if
    end for
    for each $(C, \gamma, \hat{x}) \in solutions$ **do**
      $\Pi \leftarrow \Pi \cup SVM(C, \gamma, \hat{x})$
    end for
  end for
else
  $PS \leftarrow$ Optimization$(\mathcal{LS})$
  for each $(C, \gamma, \hat{x}) \in PS$ **do**
    $\Pi \leftarrow \Pi \cup SVM(C, \gamma, \hat{x})$
  end for
end if

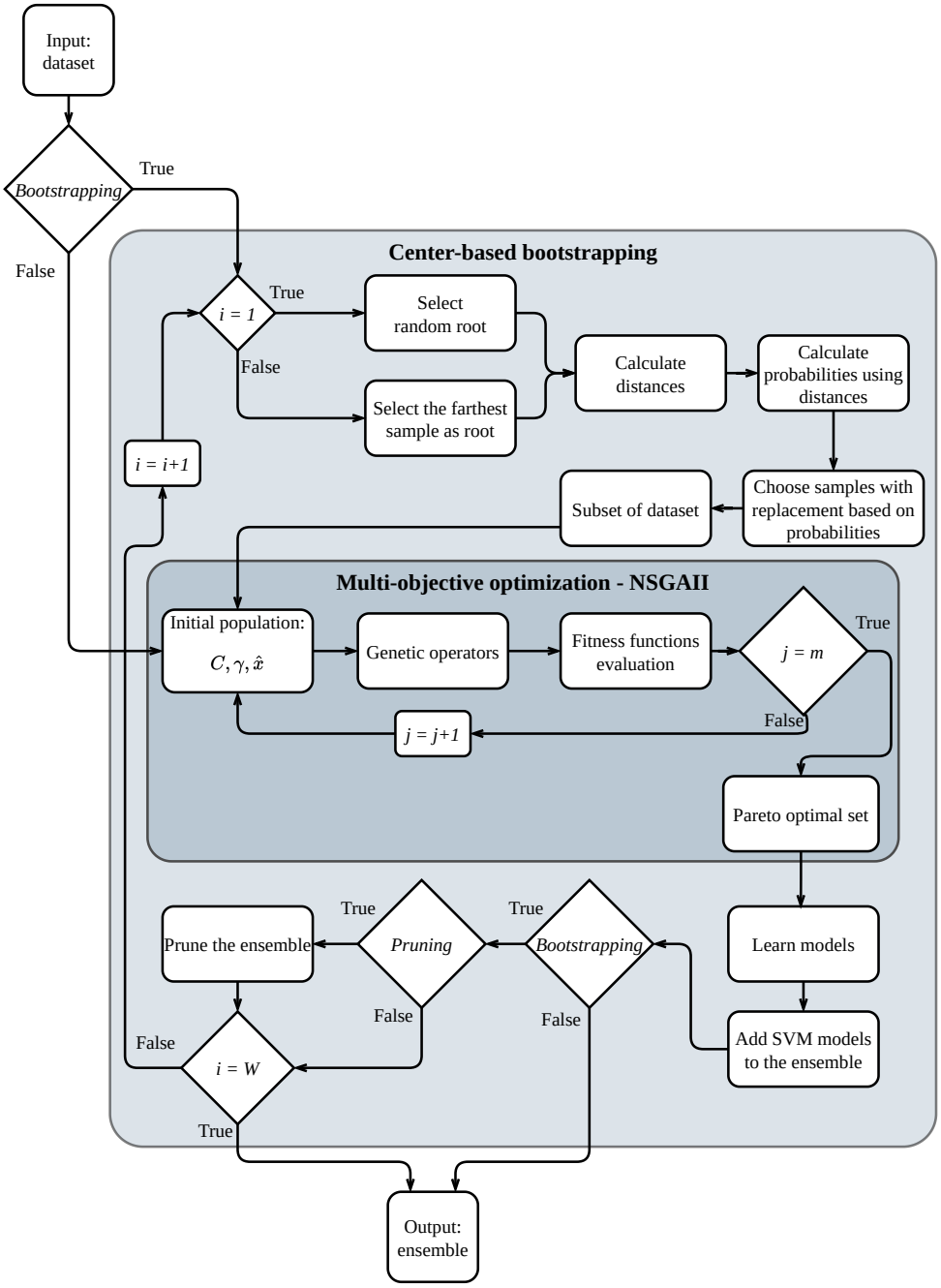
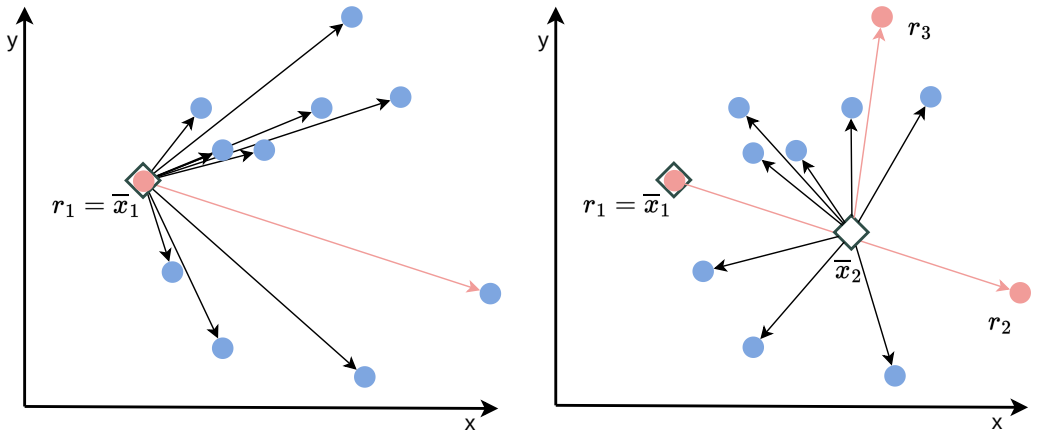


Figure 4.1: *Diagram of proposed methods: SEMOOS, SEMOOSb (when bootstrapping is used), SEMOOSbp (when bootstrapping and pruning are applied)*

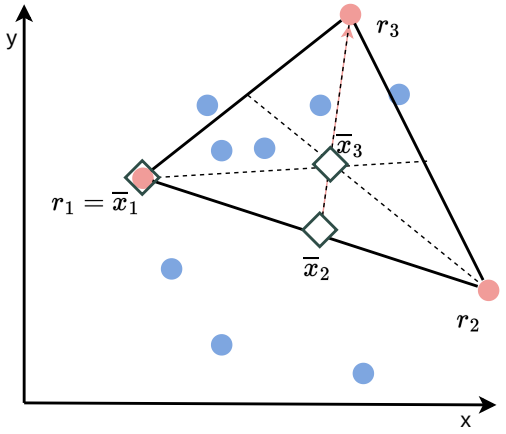
$$center(x_c, k, x) = \frac{1}{k+1} \begin{bmatrix} kx_c^{(1)} + x^{(1)} \\ kx_c^{(2)} + x^{(2)} \\ \vdots \\ kx_c^{(d)} + x^{(d)} \end{bmatrix} \tag{4.1}$$

An example of the algorithm's operation in two-dimensional space is shown and described in Fig. 4.2.



(a) *The randomly selected root r_1 is the center $\overline{x}_1$. The next root r_2 is the furthest example from the center $\overline{x}_1$.*

(b) *The middle point between r_1 and r_2 is the new center $\overline{x}_2$, and the furthest example from it, is the root r_3.*



(c) *After connecting three roots, the triangle is created, and its centroid is the new center $\overline{x}_3$.*

Figure 4.2: *Center-based Bootstrapping in 2D space. The blue dots are examples, and the arrows from $\overline{x}_1$ or $\overline{x}_2$ to examples are distances.*

In each iteration, the subset S_i is used as the input to the multi-objective optimization fulfilled by the *NSGA-II* algorithm [35]. Equation 4.2 presents two fitness functions, F_1 and F_2 that the optimization algorithm uses to maximize *Precision* and *Recall*

$$\begin{cases} \text{maximize } F_1(C, \gamma, \hat{x}) = Precision \\ \text{maximize } F_2(C, \gamma, \hat{x}) = Recall \end{cases} \tag{4.2}$$

The metrics are calculated during the validation process inside the optimization with the base estimator *SVM* and different values of its hyperparameters C and γ that have various lower and upper limits in the set of real numbers: $C \in$ [1e 6, 1e 9] and $\gamma \in$ [1e-7, 1e-4]. $\hat{x}$ is a binary vector containing selected features. These three parameters $(C, \gamma, \hat{x})$ form an initial population. The optimization is repeated until the maximum number of evaluations (m) is reached. It returns the non-dominated solution set containing results such as C, γ, and selected features.

Then, the estimator is trained and added to the ensemble for each result. The version *SEMOOSbp* with the *Pruning* does not add all models to the ensemble. It finds unique values of fitness functions (*Precision* and *Recall*) and, based on these indexes of solutions, trains the estimator on $(C, \gamma, \hat{x})$. Finally, the algorithm returns the ensemble of classifiers. The prediction is based on *Average Support Vectors*.

This method depends on several parameters, but we distinguish two main versions of *SEMOOS* – *SEMOOSb* and *SEMOOSbp*. *SEMOOSb* employs an original bootstrapping method to increase the diversity of the ensemble (*Bootstrapping* is true). *SEMOOSbp* also employs pruning to remove similar models from the ensemble (*Pruning* is true). The basic version of *SEMOOS* is much simpler. It starts from the multi-objective optimization described above, but the input is $\mathcal{LS}$. Subsequently, solutions from the non-dominated solution set PS such as $(C, \gamma, \hat{x})$ are used to train the model, and all models are added to the ensemble.

Computational complexity analysis

The computational complexity depends on a few aspects of the proposed method. Firstly, the time complexity of the base classifier *SVM* is $O(N^3)$ [2], where N denotes the size of the dataset, so it is the number of examples in the learning set. Let us assume that M is the number of objectives and U is the population size. The computational complexity of *NSGA-II* is $O(MU^2)$ [35]. In our case, $M = 2$, the complexity is $O(2U^2)$. The last part of the method is *Bootstrapping*, and its complexity is given by $O(N)$. The complexity of each possible step of *SEMOOS* is $O(N^3) + O(MU^2) + O(N) = O(N^3) + O(MU^2)$, and the highest of them is the overall computational complexity of all versions of the proposed methods.

4.2 Experimental evaluation

The experiments described in this section are used to test the proposed methods and answer the research questions posed below.

RQ1: **What is the impact of the *SEMOOS*'s parameters (especially *Bootstrapping* and *Pruning*) on its quality?**

RQ2: **How do variants of the *SEMOOS* affect classification quality?**

RQ3: **Can *SEMOOS* methods outperform state-of-the-art algorithms?**

RQ4: **What is diversity of *SEMOOS* ensemble compared to the reference methods?**

4.2.1 Setup

Experiments were prepared using the *Python* programming language and a few libraries: *Pymoo* [14], *scikit-learn* [114], *Numpy* [109], *Matplotlib* [67], *Pandas* [149]. The implementation of the experiments is available in the *GitHub repository*[1].

As the proposed method aims to train an ensemble of *SVM*s, the *Random Subspace* method is also based on *SVM*, which randomly performs feature selection for base classifiers. By choosing the right features for each base classifier, diversity is assured. The other reference algorithms do not use the classifier ensemble paradigm, i.e., simple *SVM* classifiers without feature selection and two *SVM* models based on feature selection. Below is a description and parameters of the *SEMOOS* algorithm's variants and the reference methods used in the experiments.

- *SEMOOS – SVM Ensemble with Multi Objective Optimization Selection* (*SVM* is the base classifier, the optimization algorithm *NSGA-II* has 100 populations, use genetic operators *Random Sampling*, *Polynomial Mutation* and *Bitflip Mutation*, *Simulated Binary Crossover* and *Two Point Crossover*; a constraint's representation $\hat{x}$ is 75%, number of evaluations is 1000, the ensemble size is 10)
- *SEMOOSb – SVM with Multi Objective Optimization Selection with Bootstrapping* (*SVM* is the base classifier, the optimization algorithm *NSGA-II* has 100 populations, use genetic operators *Random Sampling*, *Polynomial Mutation* and *Bitflip Mutation*, *Simulated Binary Crossover* and *Two Point Crossover*; a constraint's

[1] `https://github.com/w4k2/SEMOOS_cv`

representation $\hat{x}$ is 75%, number of evaluations is 1000, the ensemble size 10 is multiplied by the number of iterations (5 iterations) of the *Bootstrapping*, and the final ensemble consists of 50 classifiers)

- *SEMOOSbp – SVM Ensemble with Multi Objective Optimization Selection with Bootstrapping and Pruning* (*SVM* is the base classifier, the optimization algorithm *NSGA-II* has 100 populations, use genetic operators *Random Sampling*, *Polynomial Mutation* and *Bitflip Mutation*, *Simulated Binary Crossover* and *Two Point Crossover*; a constraint's representation $\hat{x}$ is 75%, number of evaluations is 1000, the ensemble size 10 is multiplied by the number of iterations (5 iterations) of the *Bootstrapping*, and the ensemble consists of 50 classifiers, but when using *Pruning* its size cannot be specify in advance)

- *RS – Random Subspace SVM Ensemble* [64] (the number of classifiers is 100 and the number of subspaces is 3 yield the output shape of subspaces, the base classifier is *SVM*)

- *SVM – Support Vector Machines* [132] (the regularization parameter $C = 1.0$, the kernel is *RBF*, kernel coefficient γ scaled, probability estimates are set to $True$)

- *FS – Feature Selection SVM* (uses *Chi-square* statistic and the *K-best* function, where K is 75% of features)

- *FSIRSVM – Feature Selection Imbalance Ratio SVM* (uses *Chi-square* statistic and the *K-best* function, where K is 75% of features, the class weight parameter of *SVM* is *IR*)

Three versions of the proposed method (*SEMOOS*, *SEMOOSb*, *SEMOOSbp*) will be compared with the following benchmark solutions: *SVM*, an ensemble *RS*, and two classifiers with feature selection (*FS* and *FSIRSVM*). *FS* is a classical approach to Feature Selection. It is based on the *Chi-square* statistic [135] and the *K-best* function, which chooses *K-best* features. *FSIRSVM* is almost the same as *FS*, but it has an additional parameter, *IR – Imbalance Ratio* of each fold, which is applied to the *SVM* as the class weight parameter. *FS* and *FSIRSVM* select 75% of features from each dataset. *RS* creates random subspaces, and it has 100 models in the ensemble. The *Support Vector Machines* (*SVM*) classifier with the default parameters was used as a base classifier for all methods except all variants of *SEMOOS*.

All *SEMOOS* variants use *SVM* and optimize its parameters C and γ. The rest of the parameters of *SVM* are default. Our methods use the optimization algorithm *NSGA-II* described in Sec. 2.3 by 100 populations with the diverse representation, because C and

γ are real values, and $\hat{x}$ is a binary vector of selected features. After pre-experiments, genetic operators were used, such as *Random Sampling*, *Polynomial Mutation* (for the real representation) and *Bitflip Mutation* (for the binary representation), *Simulated Binary Crossover* (for the real representation), and *Two Point Crossover* (for the binary representation). The *eta* parameter of *Simulated Binary Crossover* and *Polynomial Mutation* was set to 5. A constraint's representation $\hat{x}$ is 75%, meaning that 75% of features are selected and their value in the vector is 1. The process of optimizing parameters is done by 1000 evaluations. The size of the ensemble depends on the variant of our methods. The parameter determines the ensemble size in the *SEMOOS* methods, which is ten models. In *SEMOOSb*, this value is multiplied by the number of iterations (five iterations) of the *Bootstrapping*, and the final ensemble consists of 50 classifiers. Due to the *Pruning* in *SEMOOSbp*, we cannot specify its size in advance.

As the experimental protocol, the 5×2 *CV* described in Sec. 2.2.3 was chosen. Such cross-validation was also applied inside the optimization to avoid overfitting. Results were saved as *csv* files for each dataset fold and metric. Then, we performed *Wilcoxon statistical rank-sum tests* to see if one method was statistically significantly better than the other in pairwise rankings. Five metrics were used to measure the quality of methods: *BAC – Balanced Accuracy*, *Gmean*, *Recall*, and *Precision*.

All imbalanced datasets used in the experiments are listed in 4.1, where ID — is the dataset identifier, *Dataset* — is the name of the dataset, IR – is the *Imbalance Ratio*, $Ex.$ – is the number of examples, *Attr.* – is the number of attributes. They were loaded from the *KEEL dataset repository* [142], where these datasets were divided according to *IR*. The first 19 datasets are separated with the line and have *IR* lower than 9. The rest of the datasets have *IR* higher than 9. We keep up this division in our results to check the effectiveness of our methods on data with low and high imbalances. Most datasets are multi-class problems but have already been prepared as binary classification problems. For example, the *glass-0-1-2-3_ vs_ 4-5-6* dataset combines $0, 1, 2, 3$ classes of the original dataset as a negative class (majority) and $4, 5, 6$ as a positive class (minority). Fernandez et al. [49, 50] characterizes the extended description with class names.

4.3 Experiments

We carried out four groups of experiments to answer the research questions:

- Selection of the best hyperparameters of the *SEMOOS* algorithm (Experiment 1).

Table 4.1: *Description of datasets*

ID	DATASET	IR	EX.	ATTR.	ID	DATASET	IR	EX.	ATTR.
1	*glass1*	1.82	214	9	40	*ecoli-0-1-4-7_vs_5-6*	12.28	332	6
2	*wisconsin*	1.86	683	9	41	*cleveland-0_vs_4*	12.62	177	13
3	*pima*	1.87	768	8	42	*ecoli-0-1-4-6_vs_5*	13.00	280	6
4	*iris0*	2.00	150	4	43	*shuttle-c0-vs-c4*	13.87	1829	9
5	*glass0*	2.06	214	9	44	*yeast-1_vs_7*	14.30	459	7
6	*yeast1*	2.46	1484	8	45	*glass4*	15.46	214	9
7	*haberman*	2.78	306	3	46	*page-blocks-1-3_vs_4*	15.86	472	10
8	*vehicle2*	2.88	846	18	47	*abalone9-18*	16.40	731	8
9	*vehicle1*	2.90	846	18	48	*dermatology-6*	16.90	358	34
10	*vehicle3*	2.99	846	18	49	*zoo-3*	19.20	101	16
11	*glass-0-1-2-3_vs_4-5-6*	3.20	214	9	50	*glass-0-1-6_vs_5*	19.44	184	9
12	*vehicle0*	3.25	846	18	51	*shuttle-6_vs_2-3*	22.00	230	9
13	*new-thyroid1*	5.14	215	5	52	*glass5*	22.78	214	9
14	*newthyroid2*	5.14	215	5	53	*yeast-2_vs_8*	23.10	482	8
15	*segment0*	6.02	2308	19	54	*lymphography-normal-fibrosis*	23.67	148	18
16	*glass6*	6.38	214	9	55	*car-good*	24.04	1728	6
17	*yeast3*	8.10	1484	8	56	*car-vgood*	25.58	1728	6
18	*ecoli3*	8.60	336	7	57	*kr-vs-k-zero-one_vs_draw*	26.63	2901	6
19	*page-blocks0*	8.79	5472	10	58	*kr-vs-k-one_vs_fifteen*	27.77	2244	6
20	*ecoli-0-3-4_vs_5*	9.00	200	7	59	*yeast4*	28.10	1484	8
21	*yeast-2_vs_4*	9.08	514	8	60	*poker-9_vs_7*	29.50	244	10
22	*ecoli-0-6-7_vs_3-5*	9.09	222	7	61	*kddcup-guess_passwd_vs_satan*	29.98	1642	41
23	*ecoli-0-2-3-4_vs_5*	9.10	202	7	62	*abalone-3_vs_11*	32.47	502	8
24	*yeast-0-3-5-9_vs_7-8*	9.12	506	8	63	*yeast5*	32.73	1484	8
25	*yeast-0-2-5-7-9_vs_3-6-8*	9.14	1004	8	64	*kr-vs-k-three_vs_eleven*	35.23	2935	6
26	*yeast-0-2-5-6_vs_3-7-8-9*	9.14	1004	8	65	*ecoli-0-1-3-7_vs_2-6*	39.14	281	7
27	*ecoli-0-4-6_vs_5*	9.15	203	6	66	*abalone-17_vs_7-8-9-10*	39.31	2338	8
28	*ecoli-0-1_vs_2-3-5*	9.17	244	7	67	*abalone-21_vs_8*	40.50	581	8
29	*ecoli-0-2-6-7_vs_3-5*	9.18	224	7	68	*yeast6*	41.40	1484	8
30	*glass-0-4_vs_5*	9.22	92	9	69	*kddcup-land_vs_portsweep*	49.52	1061	41
31	*ecoli-0-3-4-6_vs_5*	9.25	205	7	70	*kr-vs-k-zero_vs_eight*	53.07	1460	6
32	*ecoli-0-3-4-7_vs_5-6*	9.28	257	7	71	*poker-8-9_vs_6*	58.40	1485	10
33	*yeast-0-5-6-7-9_vs_4*	9.35	528	8	72	*shuttle-2_vs_5*	66.67	3316	9
34	*vowel0*	9.98	988	13	73	*abalone-20_vs_8-9-10*	72.69	1916	8
35	*ecoli-0-6-7_vs_5*	10.00	220	6	74	*kddcup-buffer_overflow_vs_back*	73.43	2233	41
36	*ecoli-0-1-4-7_vs_2-3-5-6*	10.59	336	7	75	*kddcup-land_vs_satan*	75.67	1610	41
37	*led7digit-0-2-4-5-6-7-8-9_vs_1*	10.97	443	7	76	*kr-vs-k-zero_vs_fifteen*	80.22	2193	6
38	*ecoli-0-1_vs_5*	11.00	240	6	77	*poker-8_vs_6*	85.88	1477	10
39	*glass-0-6_vs_5*	11.00	108	9	78	*kddcup-rootkit-imap_vs_back*	100.14	2225	41

- Comparison of the main variants of the proposed method, i.e., investigation of the effect of bootstrapping and ensemble pruning on the quality of *SEMOOS* (Experiment 2).
- Comparison of the variants of the proposed method with selected reference algorithms (Experiment 3).

- Evaluation of classifier ensemble diversity of the proposed variants of *SEMOOS* (Experiment 4).

4.3.1 Experiment 1: Setting hyperparameters

Our methods have a few parameters, so we conducted pre-experiments to choose values of these parameters that get the best quality. We tested *SEMOOSb* on four datasets (*ecoli-0-3-4-7_vs_5-6*, *glass5*, *vehicle3*, *yeast-2_vs_4*) and averaged results to present figures.

Firstly, we check the *eta* parameter of crossover and mutation for each metric for values $[2, 5, 10, 20]$. Deb et al. in [36] point out that small *eta* values for crossover provide a diverse search among solutions. Fig. 4.3 shows exemplary results for one dataset *yeast-2_vs_4*, where *eta_m* and *eta_c* are *eta* parameters for mutation and crossover accordingly. Figures presenting the rest datasets are available in the *GitHub repository*[2] [3]. The best results is a black square with bold, white font inside. After analyzing all datasets, the results shown in Fig. 4.3 proved that the parameters achieving the highest *BAC*, *Gmean*, and *Recall* are $eta\_c = 5$ and $eta\_m = 5$. *Precision* is not the highest for these values, but the metric equal 0.916 are not much worse.

Next, the number of iterations of bootstraps and the percent of selected features were tested, and *ecoli-0-3-4-7_vs_5-6* dataset is shown in Fig. 4.4. This situation is similar to the previous one, *BAC*, *Gmean*, and *Recall* indicate the highest metrics for $bootstrap = 5$ and $features = 75\%$.

4.3.2 Experiment 2: Comparison of three variants of *SEMOOS*

Based on the parameters selected in the previous experiment, tests are conducted comparing the proposed *SEMOOS*, *SEMOOSb*, and *SEMOOSbp* methods. The results from the folds and all datasets are averaged, and *Wilcoxon rank-sum statistical tests* are performed. In the figures of the Wilcoxon test (Fig. 4.5, 4.6), rows are different metrics (*BAC*, *Gmean*, *Recall*, *Precision*), and columns are labeled with methods. The green color means that the method wins, yellow – ties, and red – loses to the method located at the bar on the chart. The black dashed line indicates the statistical significance of the method as winning. Fig. 4.5 shows a test for datasets with the *Imbalance Ratio* of less than 9. It can be noticed that *SEMOOS* and *SEMOOSbp* win with statistical

[2] `https://github.com/w4k2/SEMOOS_cv/tree/main/results/experiment0_set_crossmut/grid/plots`

[3] `https://github.com/w4k2/SEMOOS_cv/tree/main/results/experiment0_set_featboot/grid/plots`

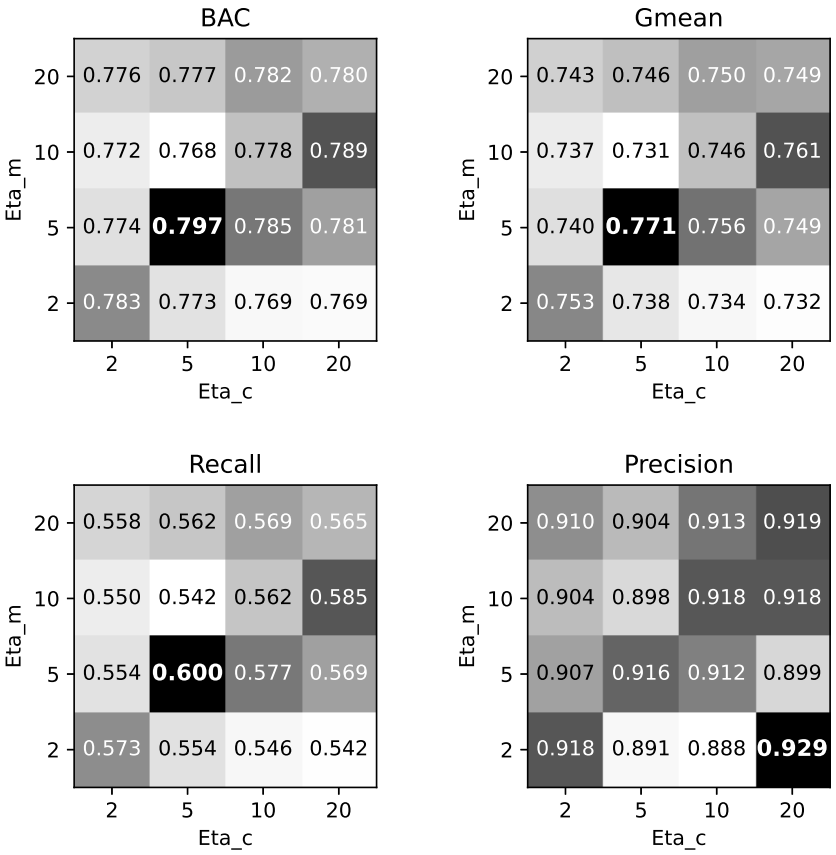


Figure 4.3: *First pre-experiment: setting eta parameter for mutation and crossover (the best result – the black square)*

significance with the *SEMOOSb* method, especially in the first three metrics, i.e., *BAC*, *Gmean*, and *Recall*. In Fig. 4.6, there are 59 datasets where $IR > 9$. Similar conclusions can be drawn from it that *SEMOOS* is statistical significance better than *SEMOOSb*, but the other methods are similar.

4.3.3 Experiment 3: Comparison with reference methods

A vital element of each method is to compare it with state-of-the-art methods to check whether the proposed new method is statistically significantly better than the others. This experiment will compare three proposed variants of the *SEMOOS* method and the reference methods *RS*, *SVM*, *FS*, *FSIRSVM*. In the *GitHub repository*[4][5], there are tables with the exact results of the metrics averaged over the folds together with the standard deviation for each dataset. The best result for a given dataset is marked in bold. Each

[4] `https://github.com/w4k2/SEMOOS_cv/tree/main/results/experiment_server/experiment5_cross_val_in_opt_9l/raw_results`

[5] `https://github.com/w4k2/SEMOOS_cv/tree/main/results/experiment_server/experiment5_cross_val_in_opt_9h/raw_results`

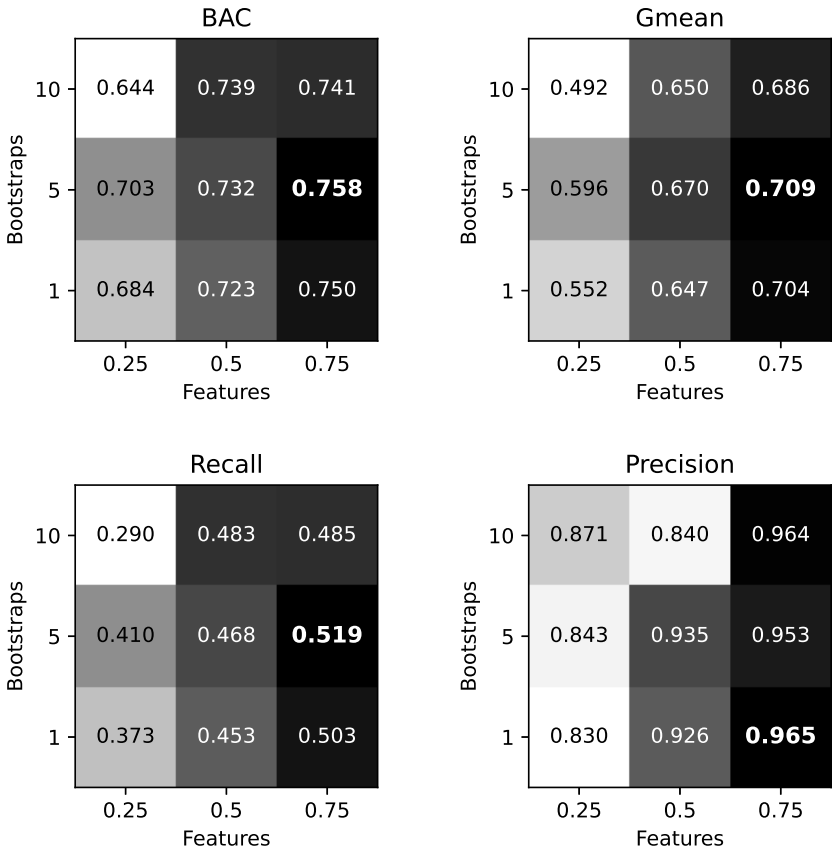


Figure 4.4: *Second pre-experiment: setting iteration of Bootstrapping and the percent of selected features (the best result – the black square)*

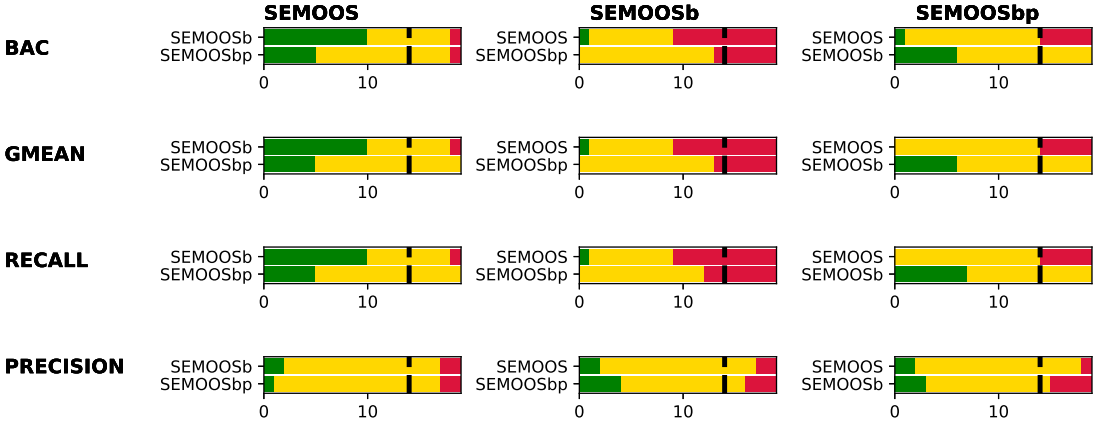


Figure 4.5: *Wilcoxon rank-sum test of proposed methods for datasets with $IR < 9$ (green – win, yellow – tie, red – loss)*

table contains a different metric. The results of the proposed methods are compared with the reference ones. Sometimes, the quality is slightly higher or lower, but it is not easy to pinpoint a winning method.

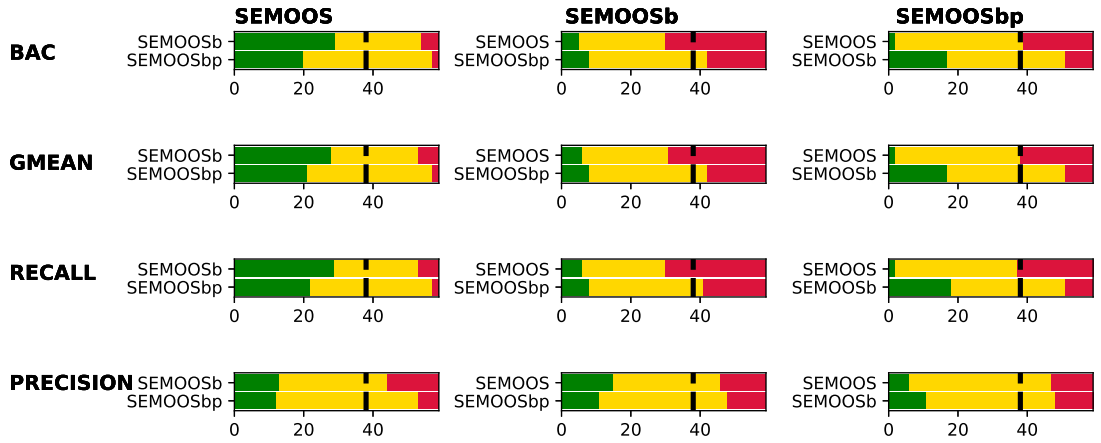


Figure 4.6: *Wilcoxon rank-sum test of proposed methods for datasets with $IR > 9$ (green – win, yellow – tie, red – loss)*

Therefore, *Wilcoxon statistical tests* showed how many datasets each proposed method won. As in the previous section, the figures are grouped by *Imbalance Ratio* above or below 9. Each *SEMOOS* variant is compared to all reference methods for the five metrics. All *SEMOOS* variants work similarly in Fig. 4.7, and they are statistically significantly better than *RS* and *FSIRSVM* for *BAC*, *Gmean*, and *Recall* metrics. They also achieved many wins compared to the *RS* method. However, only *SEMOOS* for the *Recall* metric shows a statistically significant win.

The results in Fig. 4.8 differ slightly for data with high imbalance. There are many more datasets in this case. All variants of the *SEMOOS* method win with static significance with the *RS* method for *BAC*, *Gmean*, and *Recall* metrics. There are also fewer draws between the methods for these three metrics. For the *Precision* metric, all variants of the *SEMOOS* method win with statistical significance over the *RS* method and tie with the remaining ones on many datasets. An important goal for us was to correctly identify the minority class that is represented by *Recall*.

Analysis of the non-dominated solution set in the objective functions space is presented in Fig. 4.9. Triangles represent the reference methods. Our method proposals fall into two categories: the method name with *PF* (non-dominated set), and the method name alone is the quality of the final constructed ensemble. These scatter plots show how the non-dominated solutions are located relative to the final built ensembles and the reference methods. Fig. 4.9a shows an exemplary broad non-dominated set, which provides more diverse models. The reference methods are behind the non-dominated solutions; when comparing them to ensembles, they obtain a lower *Precision* value. Fig. 4.9b shows more focused non-dominated solutions than Fig. 4.9a. *SEMOOS* variants obtain a similar value for both metrics and win over the reference methods. In these

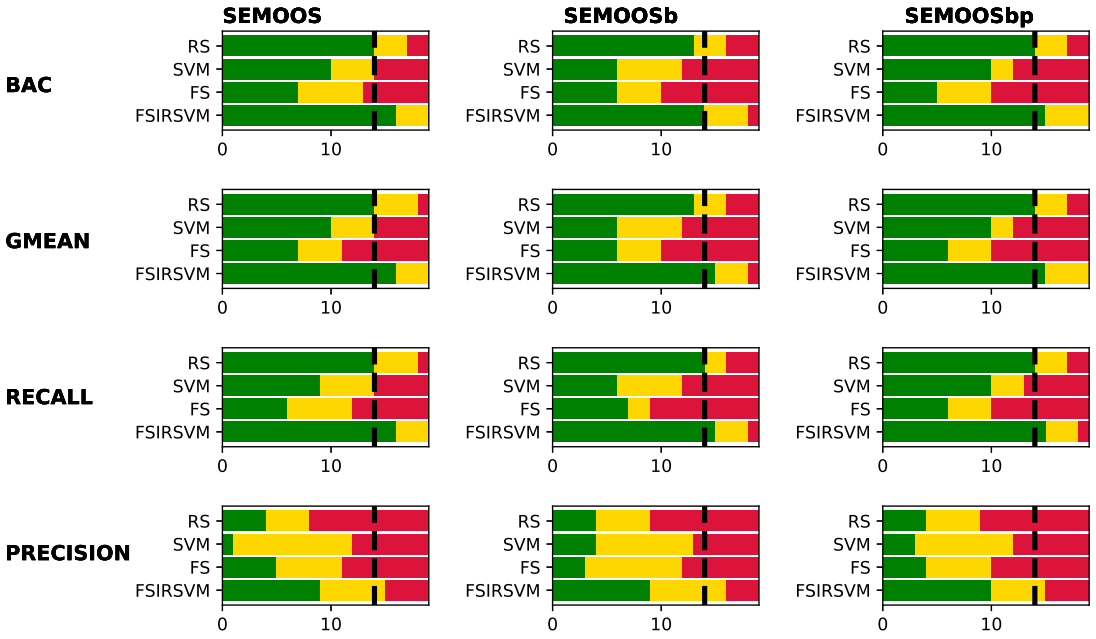


Figure 4.7: *Wilcoxon rank-sum test for datasets with $IR < 9$ (green – win, yellow – tie, red – loss)*

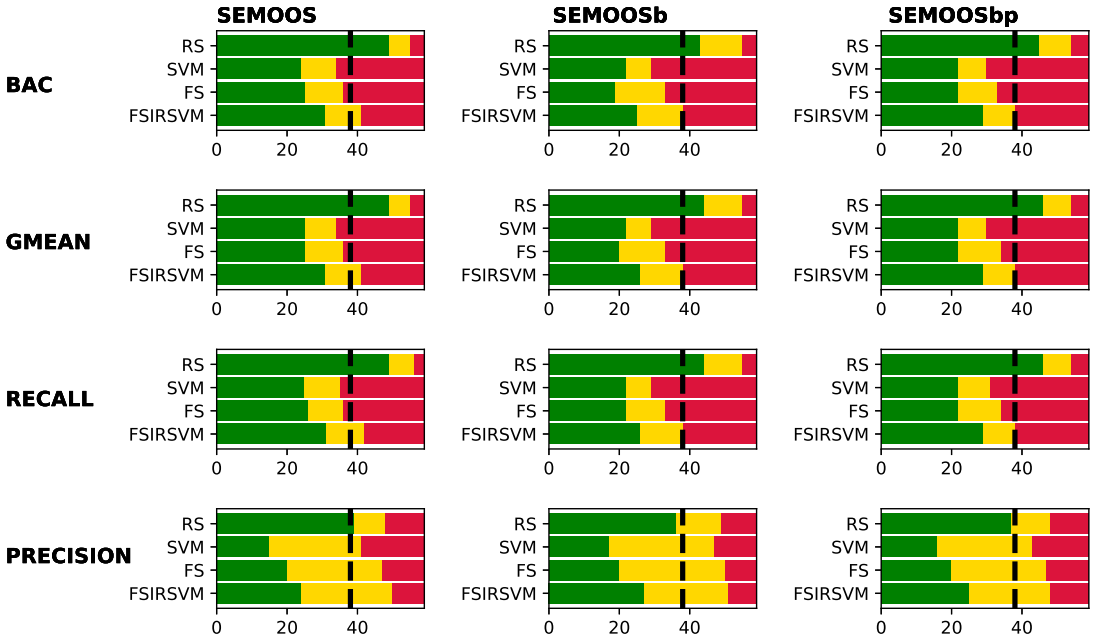
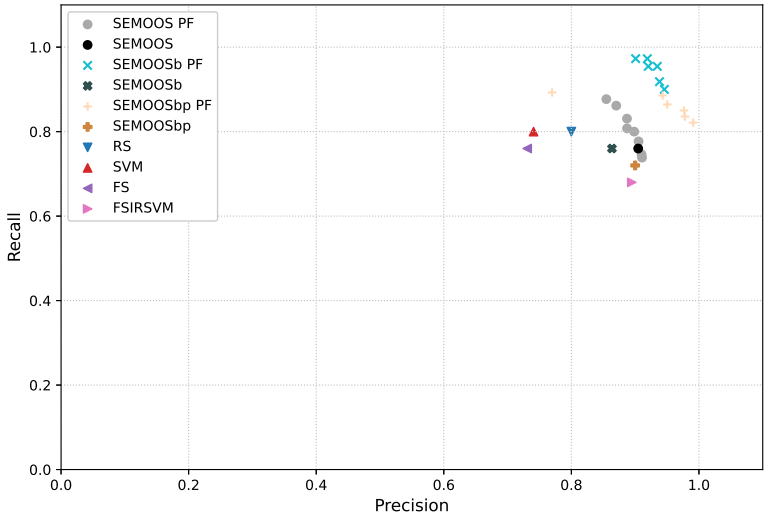
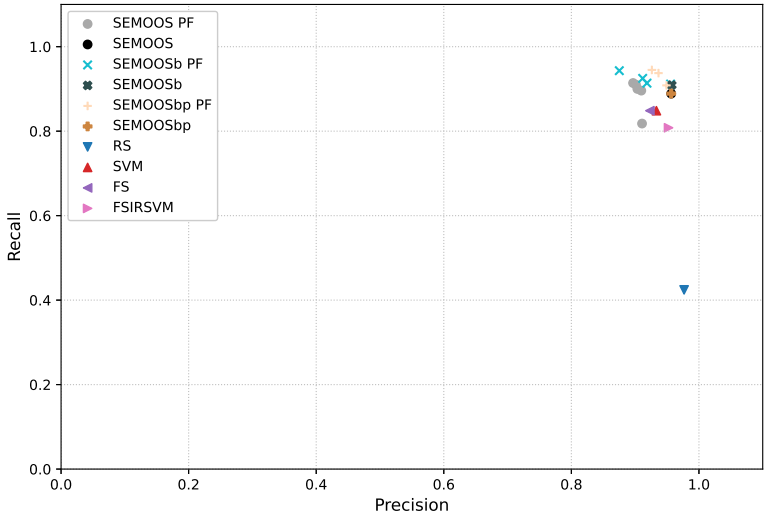


Figure 4.8: *Wilcoxon rank-sum test for datasets with $IR > 9$ (green – win, yellow – tie, red – loss)*

examples, the methods score is both high *Precision* and *Recall*, but this is not a rule for all datasets.



(a) Dataset glass-0-1-2-3_ vs_ 4-5-6



(b) Dataset vehicle0

Figure 4.9: *Scatter plots with non-dominated solutions and reference methods*

4.3.4 Experiment 4: Evaluating *SEMOOS* ensemble diversity

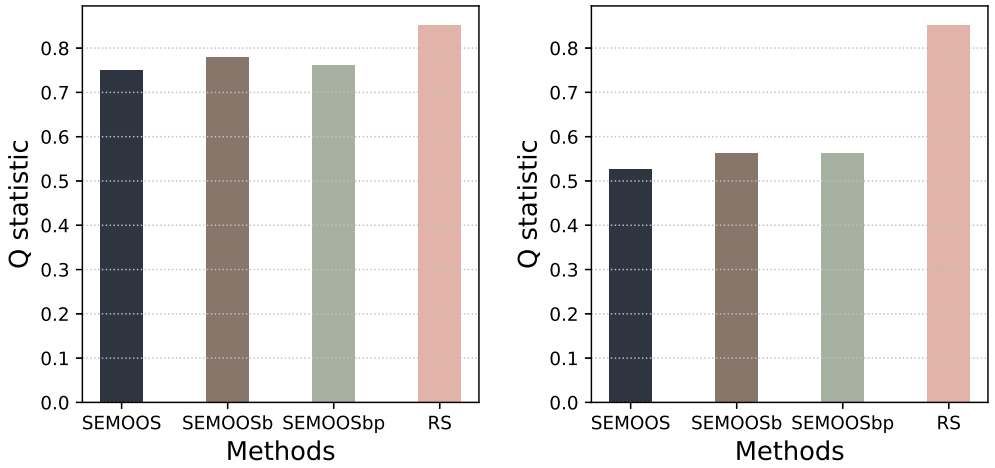
The last experiment focuses on evaluating the diversity of models in the ensemble. We conducted tests for four diversity metrics proposed in [82] for three *SEMOOS* variants and one reference ensemble *RS*. Fig. 4.10 shows the *Q-statistic* metric, while all other metrics are available on the remote *GitHub repository*[6] [7].

[6] `https://github.com/w4k2/SEMOOS_cv/tree/main/results/experiment_server/experiment5_cross_val_in_opt_9h/diversity_plot`

[7] `https://github.com/w4k2/SEMOOS_cv/tree/main/results/experiment_server/experiment5_cross_val_in_opt_9l/diversity_plot`

Q-statistic is a pairwise diversity measure. It can assess two classifier outputs and return the decision on their similarity. Q is in the range from -1 to 1, where $Q = 0$ means that classifiers are statistically independent; $Q < 0$ – classifiers make mistakes on different objects; $Q > 0$ – classifiers correctly recognize the same objects.

From Fig. 4.10a, it may be concluded that the method diversity hardly differs for data with a slight imbalance. However, these differences are significant for a more significant imbalance (Fig. 4.10b). The result of the *SEMOOS* method is closest to the value 0, which means that the models in this method are the most diverse, compared to *RS*, the difference is 0.3.



(a) *Datasets with $IR < 9$* ***(b)*** *Datasets with $IR > 9$*

Figure 4.10: *Diversity measure: Q statistic*

4.4 Lessons learned

Let us try to answer the research questions considering the results obtained from the experiments.

RQ1: **What is the impact of the *SEMOOS*'s parameters (especially *Bootstrapping* and *Pruning*) on its quality?**

During pre-experiments, four parameters were examined: *eta_c* for crossover and *eta_m* for mutation, the number of Bootstrap iterations, and the number of features. The quality of the *eta* parameter for both genetic operators depends on the selected metric. The most significant difference between the best and worst results is 0.3, but it is usually only a few hundredths. Combining different values of the *eta_c* and *eta_m* parameters does not give unequivocal results. The analysis of

these cases showed that good results were obtained for the values of $eta\_c = 5$ and $eta\_m = 5$. It is easier to draw the following conclusions for the other two parameters. The greater the number of features selected, the higher the value of most metrics. The results are the worst for the 1 and 10 iterations of *Bootstrapping*. Therefore, we selected 75% of the features and five iterations of *Bootstrapping* for further experiments to obtain the highest classification quality. *Bootstrapping* and *Pruning* as parameters of the *SEMOOS* method improve the quality of the tested metric.

RQ2: **How do variants of the *SEMOOS* method affect classification quality?**
None of the proposed methods shows a statistically significant difference compared to the others. Therefore, we included all three variants in further experiments. However, based on the presented statistical test results, *SEMOOS* and *SEMOOSbp* are better than the *SEMOOSb* for all datasets.

RQ3: **Can *SEMOOS* methods outperform state-of-the-art algorithms?**
Each variant of the *SEMOOS* method has been compared with four state-of-the-art methods, and the statistical test results for each *SEMOOS* variant are similar if we consider the number of wins. However, depending on the level of dataset imbalance, the results vary. For datasets with the *Imbalance Ratio* below 9, the proposed methods exceed the *FSIRSVM* and *RS* methods with statistical significance. However, statistical significance is obtained for the *Imbalance Ratio* above 9 for the *RS* algorithm.

RQ4: **What is diversity of *SEMOOS* ensemble compared to the reference methods?**
The last element of the research shows that all proposed methods are more diversified than the *Random Subspace* (*RS*) method. When analyzing the figure with a greater *Imbalance Ratio* with a larger number of datasets, it is noticeable that the *SEMOOS* method works best.

The main goal of this work was to use multi-objective optimization (*NSGA-II*) in the form of two independent fitness functions, *Precision* and *Recall*, to classify imbalanced data. Three classifiers ensemble using *SVM* as a base model were proposed: *SEMOOS*, *SEMOOSb*, and *SEMOOSbp*. *SEMOOS* is a basic version that adds all *SVM* models obtained from optimization. *SEMOOSb* has the additional option of *Bootstrapping*, i.e., resampling a dataset with replacement is performed to get more samples from one dataset. The last version is *SEMOOSbp*, and the method includes both the *Bootstrapping* and *Pruning* needed to remove redundant models from the ensemble. The experiments were performed according to the experimental protocol on 78 imbalanced datasets. First,

SEMOOS hyperparameters were set to select the parameter values that produce the best results for this kind of data. Then, the three versions of the *SEMOOS* method were compared with each other, but none of the methods showed statistical significance. Therefore, all versions were selected for further research. The main experiment was to compare the proposed methods with various state-of-the-art methods, such as single *SVM* classifier, classifier ensemble (*RS*), and two methods using feature selection (*FS* and *FSIRSVM*). Statistical tests showed that the *SEMOOS* variants outperform the *RS* and *FSIRSVM* methods with statistical significance. The work's last element was to compare models' diversity in ensemble methods. The presented results show that the proposed methods are more diversified than the state-of-the-art solutions.

Chapter 5

Ensemble learning on feature subspace methods

This chapter presents two ensemble classifiers built on decision tree and feature subspaces found by optimization algorithms. The first classifier uses *Differential Evolution* as a single-criteria algorithm, and the second – *MOEA/D* as a multi-objective algorithm. Each proposed approach presents a description of the method, its computational complexity, and experimental evaluation with statistical analysis. The conducted experiments allow for answering the formulated research questions summarizing the proposed methods. The questions concern the impact of hyper-parameters, optimization criteria and their number, the non-dominated solution characteristics, and the overall performance of the proposed methods compared to reference methods.

5.1 DE-Forest – optimized decision tree ensemble

Classifier ensembles are one of the most promising directions in data classification. The critical problem is to ensure an adequate diversity level of the base classifiers included in the ensemble. One of the most popular approaches is manipulating the base classifiers' input, i.e., vertical or horizontal partitioning. While horizontal partitioning does not yield good results, attribute selection is one of the most promising techniques [82]. The most popular approach is random attribute selection proposed in *Random Subspace* [64] and further developed for *Random Forest* (RF) [19], where base classifiers were decision trees (DT). Further, they are trained on bootstrapped samples to boost the diversity of the individual predictors. The selection of classifiers for a pool, i.e., the attribute selection for each model, is random, which may lead to a poor-quality ensemble in a worst-case scenario. The random nature of the described approaches may also affect

their stability. Hence, it seems an attractive approach to propose a feature selection for each base classifier in the optimization process. It guarantees a qualitatively better choice than the random search, e.g., by *Random Forest*.

Let us propose the method *DE-Forest* [57] – the ensemble consisting of optimized decision tree classifiers using *Differential Evolution*. *DE* selects the best set of decision trees that uses chosen attributes. As the fitness function, a performance metric could be used, e.g., *BAC*, *Gmean*, or *AUC*. The choice of the metric is one of the *DE-Forest* parameters. The $d \times n$ dimensional vector PS represents a feature selection. d is the number of attributes and n stands for ensemble size. Thus, the $PS[k]$ describes if the $\left(\lceil \frac{k}{d} \rceil\right)$th decision tree uses the $\left(k - \lfloor \frac{k}{d} \rfloor d\right)$th attribute.

The proposed algorithm can also employ dataset bootstrapping to increase the diversity of the returned decision trees as well as classifier ensemble pruning techniques. The use of them is also *DE-Forest* parameter.

Figure 5.1 and Algorithm 5.1 present the *DE-Forest*. Let us briefly describe the primary step of the proposed method. The ensemble uses base classifiers returned from the optimization process using the learning set $\mathcal{LS}$. If bootstrapping is disabled (step 2.), then there is additional 5×2 *CV* within the *DE* optimization. It ensures that the input data is split into training and test data to evaluate the classifiers properly. In contrast, cross-validation is no longer necessary with bootstrapping enabled (step 5.) because there are B subsets from the dataset randomly selected with replacement (step 8.).

Then, optimization using the *DE* algorithm is used (step 10). Each population in *DE* consists of $d \times n$ values. The algorithm searches for the population by crossover and evaluates it through the fitness function. Optimization runs until the maximum number of populations is reached and returns a solution (PS – a vector list). Because *DE* returns solutions represented by real numbers as shown in Figure 5.2, we have to binarize them using simple thresholding (step 13.). In step 11., j is a consecutive vector, and in step 12., m is a consecutive bit in the vector. The thresholding process changes values in the vector in such a way: if the value is greater than 0.5 for a given feature – it is selected. Otherwise, the feature is not selected. After this process, a vector consisting of *True* and *False* values is partitioned into subspaces for each model in the ensemble. A single model is trained on the feature subspace (step 15.) and added to the ensemble (step 16.). An additional option is model pruning (step 19.). The *BAC* metric is calculated for each model and then added to the list *MB* (step 20.), which is sorted descending – *MS* (step 21.). The p parameter determines the percentage of models removed from the ensemble from the end of the *MS* list. The method's output is the ensemble classifier (step 22.).

Algorithm 5.1: *DE-Forest method*

Input:

$\mathcal{LS} = \{(x_1, i_1), (x_2, i_2), \ldots, (x_N, i_N)\}$ – learning set
$TrainDT()$ – DT classifier training method based on the selected subset of features
$Bootstrapping$ – boolean parameter $Bootstrapping$

Symbols:

B – number of bootstrapped datasets
n – number of models in the ensemble
$Bootstrapping$ – boolean parameter stands for applying bootstrapping
$Pruning$ – boolean parameter stands for applying ensemble pruning

Output:

Π – pool of DT base classifiers

if not $Bootstrapping$ **then**
  $B \leftarrow 1$
end if
for $i = 1$ to B **do**
  if $Bootstrapping$ **then**
    $S_i \leftarrow$ resampling with replacement from $\mathcal{LS}$
  else
    $S_i \leftarrow \mathcal{LS}$
  end if
  $PS \leftarrow$ Optimization(S_i) ▷ PS is $(f * n)$ dimensional vector
  for $j = 1$ to n **do**
    for $m = 1$ to f **do**
      $PS[j + m - 2] \leftarrow \lceil PS[j + m - 2] - 0.5 \rceil$
    end for
    $DT \leftarrow TrainDT([PS[(j-1) * f + 1], ..., PS[j * f]]$
    $\Pi \leftarrow \Pi \cup DT(\hat{x}_j)$
  end for
end for
if Pruning **then**
  $MB \leftarrow$ calculate $metric$ for each $DT(\hat{x})$ in Π
  $MS \leftarrow$ sort MB array in descending order
  $\Pi \leftarrow \Pi - DT(\hat{x})$ ▷ Remove p% models with the worst $metric$ based on MS
end if

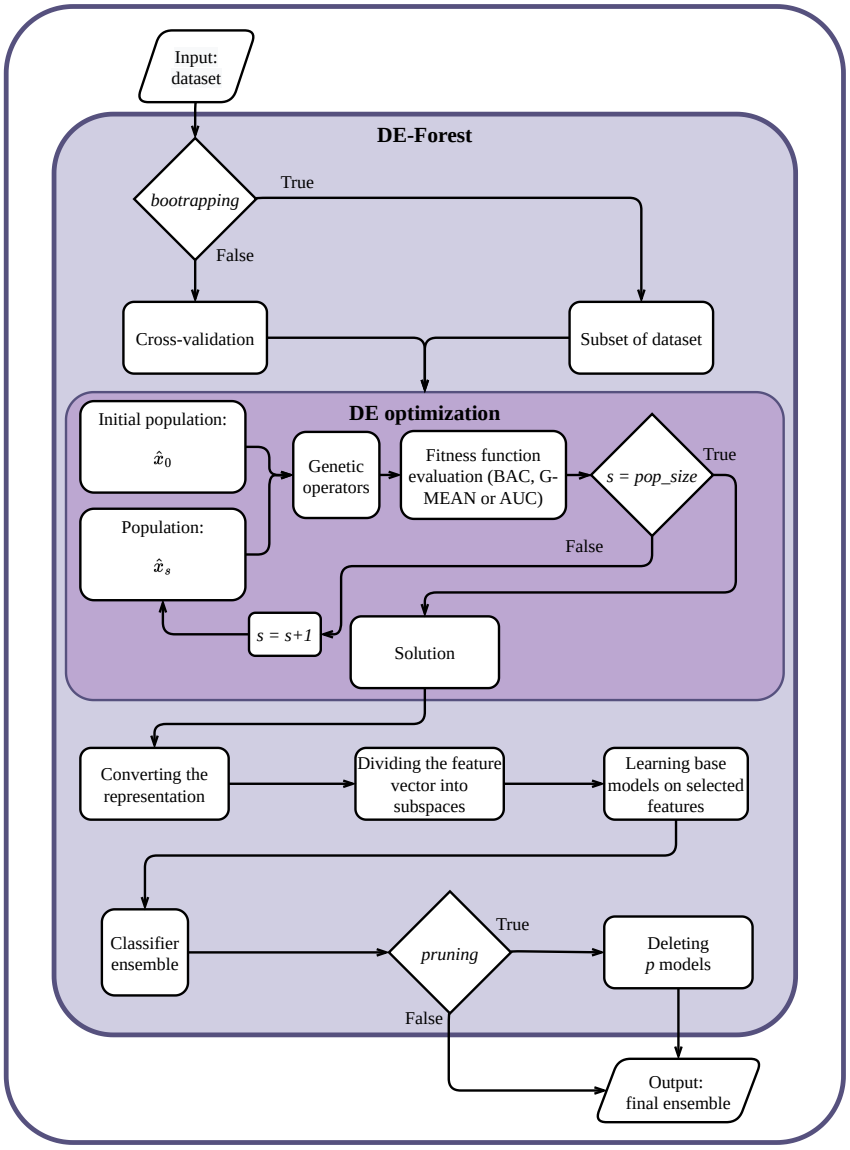


Figure 5.1: *The DE-Forest method diagram*

Computational complexity analysis

To analyze the computational complexity of the proposed method, let us first introduce the complexity of the *CART Decision Tree* – $O(dN \times log_2 N)$, where d is the number of attributes, and N is the number of samples [124]. Thus, the computational complexity of training n DTs is $O(ndN \times log_2 N)$. Additionally, since the computational complexity of *Differential Evolution* is $O(P \times dim \times G_{max})$, where P – the population size, dim – the dimension of the searched space, and G_{max} is the fixed number of generations [34]. Then, the overall computational complexity of the *DE-Forest* is $O(ndN \times log_2 N + P \times dim \times G_{max})$.

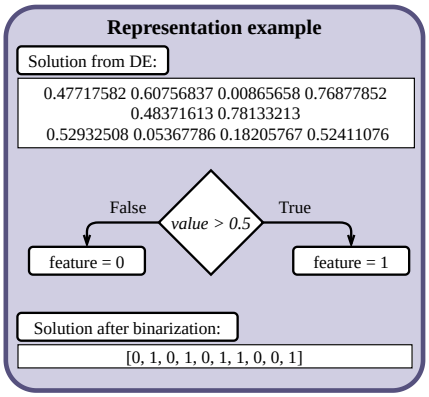


Figure 5.2: *The representation example*

5.1.1 Experimental evaluation

We performed a series of computer experiments to verify the quality of the proposed method. In the course of them, we intended to answer the following research questions:

RQ1: **What is the impact of hyperparameters on the proposed method and classification quality?**

RQ2: **How do various optimization criteria affect the *DE-Forest* method performance?**

RQ3: **Can the proposed algorithm outperform state-of-the-art methods?**

Setup

This section thoroughly overviews all the constituents needed to conduct reliable experiments. Thanks to this description, it is possible to reproduce the research.

The implementation of methods and the experimental environment are done using the *Python* programming language and a few libraries: *Pymoo* [14], *scikit-learn* [114], *Imbalanced-learn* [86], *Numpy* [109], *Matplotlib* [67], *Pandas* [149]. Complete source code, sufficient to repeat the experiments, was made available at *GitHub repository*[1]. Additionally, we provided the complete results of the experiments that were conducted.

All used datasets are obtained from *Keel-dataset repository* [4], presented in Table 5.1 sorted by the *Imbalance Ratio*. Columns in this table are as follows: *ID* – the number of datasets, *Dataset* – the name of the dataset, *IR* – *Imbalance Ratio*, *Ex.* – the number of instances, *Attr.* – the number of attributes. These are two-class problems. Datasets are

[1] `https://github.com/w4k2/DE-Forest`

used to relate to problems such as diseases, the quality of the wine, types of glass, and much more. The problem of complex, imbalanced data dictates the selection of these datasets. The selected datasets have a wide range of parameters, particularly the *IR*, and are considered benchmarks for this problem [107].

Table 5.1: *Description of datasets*

ID	DATASET	IR	EX.	ATTR.	ID	DATASET	IR	EX.	ATTR.
1	*glass1*	1.82	214	9	21	*glass-0-1-6_vs_5*	19.44	184	9
2	*vehicle3*	1.92	846	18	22	*yeast-2_vs_8*	23.10	482	8
3	*vehicle1*	1.97	846	18	23	*flare-F*	23.79	1066	11
4	*haberman*	2.78	306	3	24	*yeast4*	28.10	1484	8
5	*yeast3*	8.10	1484	8	25	*winequality-red-4*	29.17	1599	11
6	*page-blocks0*	8.79	5472	10	26	*poker-9_vs_7*	29.50	244	10
7	*yeast-2_vs_4*	9.08	514	8	27	*winequality-white-9_vs_4*	32.60	168	11
8	*ecoli-0-6-7_vs_3-5*	9.09	222	7	28	*abalone-17_vs_7-8-9-10*	39.31	2338	8
9	*glass-0-1-5_vs_2*	9.12	172	9	29	*abalone-21_vs_8*	40.50	581	8
10	*yeast-0-3-5-9_vs_7-8*	9.12	506	8	30	*yeast6*	41.40	1484	8
11	*yeast-0-2-5-6_vs_3-7-8-9*	9.14	1004	8	31	*abalone-19_vs_10-11-12-13*	49.69	1622	8
12	*ecoli-0-1_vs_2-3-5*	9.17	244	7	32	*kr-vs-k-zero_vs_eight*	53.07	1460	6
13	*ecoli-0-2-6-7_vs_3-5*	9.18	224	7	33	*ecoli1*	63.75	336	7
14	*ecoli-0-6-7_vs_5*	10.00	220	6	34	*winequality-red-3_vs_5*	68.10	691	11
15	*glass-0-1-6_vs_2*	10.29	192	9	35	*ecoli2*	70.00	336	7
16	*ecoli-0-1-4-7_vs_2-3-5-6*	10.59	336	7	36	*abalone-20_vs_8-9-10*	72.69	1916	8
17	*glass-0-1-4-6_vs_2*	11.06	205	9	37	*kddcup-buffer_overflow_vs_back*	73.43	2233	41
18	*cleveland-0_vs_4*	12.31	173	13	38	*poker-8_vs_6*	85.88	1477	10
19	*page-blocks-1-3_vs_4*	15.86	472	10	39	*kddcup-rootkit-imap_vs_back*	100.14	2225	41
20	*abalone9-18*	16.40	731	8	40	*abalone19*	129.44	4174	8

The methods, their abbreviations, and the parameters used in the experiments are presented below.

- *DT – CART Decision Tree Classifier* [20]. We used its *scikit-learn* implementation with default parameters such as:
  - *criterion of split – Gini impurity*
  - *maximum number of features – None* indicates no maximum value
- *DE-Forest – Differential Evolution Forest* ensemble with *DT* as the base classification model (the same default parameters as presented earlier). Experiment 1 set hyperparameters of *DE-Forest*: bootstrapping, metric's name, the number of classifiers, and population size. Another hyperparameter is pruning, but it was not used in the presented experiments.

- *RandomFS – Random Forest* native implementation on the basis of the original paper [19]. This implementation differs from the *RF* implementation from *scikit-learn*. One of the differences is prediction, *scikit-learn RF* combines classifiers by averaging their probabilistic prediction, but it used majority voting in the original idea. In *RandomFS*, the parameter *max features* is equal to $\sqrt{d}$, where d – the number of features. For each model, features are chosen randomly. *DT* (with the same default parameters presented earlier) has been chosen. No bootstrapping has been applied.
- *RandomFS_ b* – the method described above (*RandomFS*) with the bootstrapping option set to True.

Optimization algorithm *Differential Evolution* (*DE*) that we used in our method based on *Pymoo* implementation, with the following parameters:

- *population size* is set during experiment 1
- *sampling* is *Latin Hypercube Sampling (LHS)*
- common *variant* is *DE/rand/1/bin*, where *rand* is selection individuals to be perturbed, *1* is the number of difference vector and *bin* is the crossover type
- *Crossover Constant* is $CR = 0.9$

Methods compared during experiments based on stratified datasets using 5×2 *CV* [37].

We chose the widely used metrics to evaluate all methods: *Balanced Accuracy* (*BAC*), *Geometric Mean Score* (*Gmean*), *Recall*, *Specificity* and *Precision*. *Gmean* is based on *Precision* and *Recall* metrics.

To summarize the results for all datasets, the *Wilcoxon statistical rank-sum test* at a significance level of 0.05 was chosen [133]. The repository linked in Section 5.1.1 shows more accurate results for each dataset.

5.1.2 Results

The section presents the tuning of parameters in experiment 1 and experiment 2, comparing the proposed method and reference methods.

Experiment 1 – tuning hyperparameters using *SMAC*

Before proceeding with the main experiments comparing the proposed method and reference methods, it is necessary to select hyperparameters that will make the method obtain the best possible quality. For this purpose, the *SMAC3* library version *V2.0.0a1* was used [68, 94]. This tool uses *Bayesian Optimization* and aggressive racing mechanism to optimize parameters. It can perform optimization in many instances. In the case of the experiment, the instances are five different datasets (*glass0*, *kddcup-rootkit-imap_vs_back*, *kr-vs-k-zero_vs_eight*, *page-blocks-1-3_vs_4*, *winequality-red-8_vs_6-7*). The method hyperparameters selected for optimization are listed in Table 5.2, which includes the range of each parameter and the value after using the *SMAC* algorithm. Only one of the values listed in the *bootstrap* and *metric_name* hyperparameters are selected. *Gmean* metric is indicate as *GM*. However, ranges are given in the other two hyperparameters, and *SMAC* independently selects numbers from this range. A maximum value of 25 determines the number of classifiers in the ensemble because a larger number of models does not indicate a significant improvement in the algorithm's quality, as noted by Lin et al. in the article [93] after conducting an experiment determining the number of models in the ensemble.

Table 5.2: *Hyperparameters optimization*

Hyperparameters	**Range**	**Values**
bootstrap	$\{True, False\}$	$True$
metric_name	$\{BAC, AUC, GM\}$	BAC
n_classifiers	$[5, 25]$	15
p_size	$[100, 500]$	107

Figure 5.3 shows the dependence of the four examined hyperparameters on the BAC metric in subsequent iterations of the *SMAC* algorithm. The algorithm searches the solution space in each iteration and selects a value for each parameter. As a result of the *SMAC* algorithm, those parameter values are selected to obtain the highest BAC metric value. Each subgraph concerns one hyperparameter being optimized, the name and possible values of which are on the left y-axis. The hyperparameter value in each iteration is a stem plot – blue lines perpendicular to a baseline at each location from the baseline to heads, and a circle marker there. On the right y-axis is the BAC performance, and the solid purple line indicates the value of this metric in a given iteration. The iteration number of the *SMAC* algorithm is marked on the x-axis. In the case of this experiment, it is the seventh iteration, where $BAC = 0.858$. Comparing the second iteration, where the quality is very similar, the number of classifiers remains similar (13 classifiers), but the remaining hyperparameters have entirely different values. Hence, the

appropriate number of classifiers is significant in obtaining a high-quality metric. Other iterations did not improve the metric.

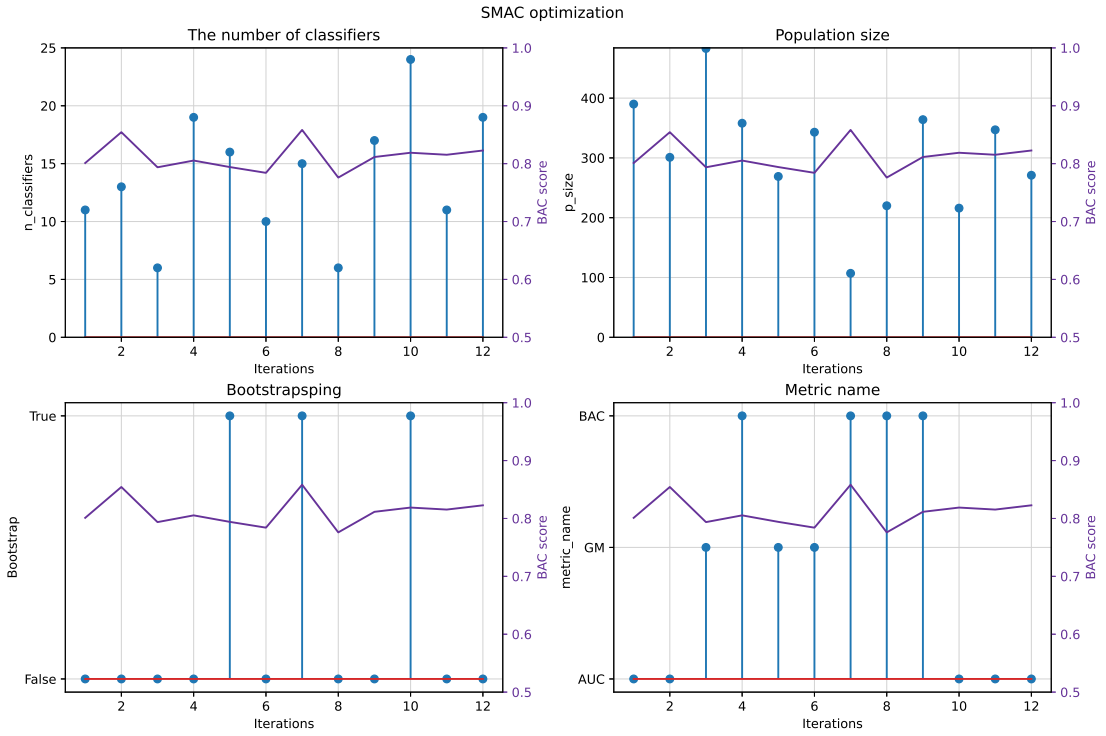


Figure 5.3: *SMAC optimization analysis*

Experiment 2 – comparative study

The main experiment aimed to compare the quality of the proposed method in the version with constraints and without, a *Random Forest* classifier (*RandomFS*) with bootstrapping and without, and (*DT*) learned on all attributes. The exact tabular results are in the repository linked in Section 5.1.1. As an illustration, the individual fold scores for each dataset were averaged and presented using the *Wilcoxon rank-sum statistical test*. The proposed method is compared with three state-of-the-art methods whose names are on the left in each small figure. Green in the figure means that the *DE-Forest* method outperformed the method in a given row, yellow means a tie, and red means a loss. The black dashed vertical line indicates the level of statistical significance. The x-axis represents the number of datasets.

Figure 5.4 shows the results for the proposed method without any constraints within the *DE* optimization. After setting the hyperparameters in the previous experiment, the *DE-Forest* method has bootstrapping set to *True*. Comparing it with *RandomFS_ b*, the *BAC*, *Specificity*, and *Precision* metrics, the method achieves better results with statistical significance. Compared to the *RandomFS* method without bootstrapping,

the proposed method wins for over half of all datasets. *DE-Forest* is better with static significance than the *DT* method for *Gmean*, *Recall*, and *Specificity* metrics.

On the other hand, high scores for some metrics of *DT* indicate favor towards the majority class, which is not desirable when classifying imbalanced data. The runtime for all methods based on this experiment is in the repository linked in Section 5.1.1. *DE-Forest* does not achieve the best times because optimization is time-consuming compared to other methods. Nevertheless, the *DE* algorithm searches for feature subspaces to build models that achieve the best possible results.

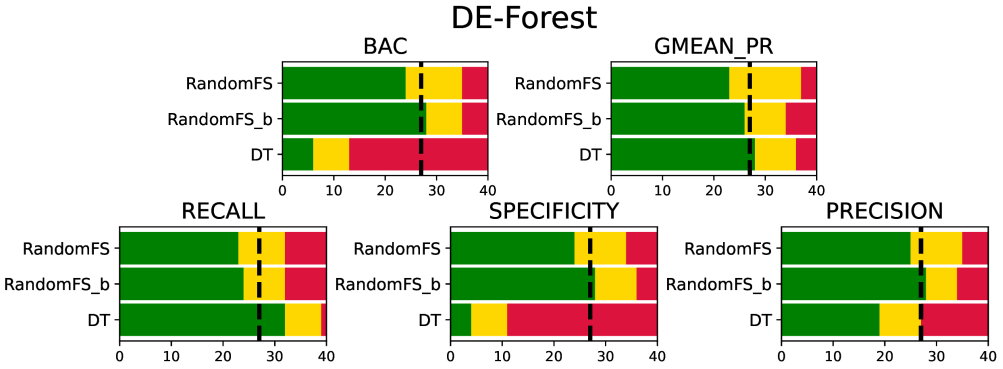


Figure 5.4: *The Wilcoxon test of DE-Forest method with reference methods (green – win, yellow – tie, red – loss)*

The second tested case is the *DE-Forest* method, which constrains the number of features to 50% of the original set of features within the *DE* optimization. Furthermore, the optimization does not search the entire subspace but is limited to half the features in advance. The advantage of this approach is the reduction of search time by the optimization algorithm, and the entire calculations are faster than *DE-Forest* without constraints. The reference methods and hyperparameters of the *DE-Forest* method are the same as in the previous case. Analyzing the individual metrics in Figure 5.5, it can be concluded that the classification quality is not much worse than the unconstrained version of the method. For the *RandomFS* and *RandomFS_b* methods, the number of wins does not fall below 20 datasets for each metric. *DE-Forest* only wins against *DT* for *Recall* with statistical significance.

5.1.3 Lesson learned

Considering the presented results, let us answer the research questions.

RQ1: **What is the impact of hyperparameters on the proposed method and classification quality?**

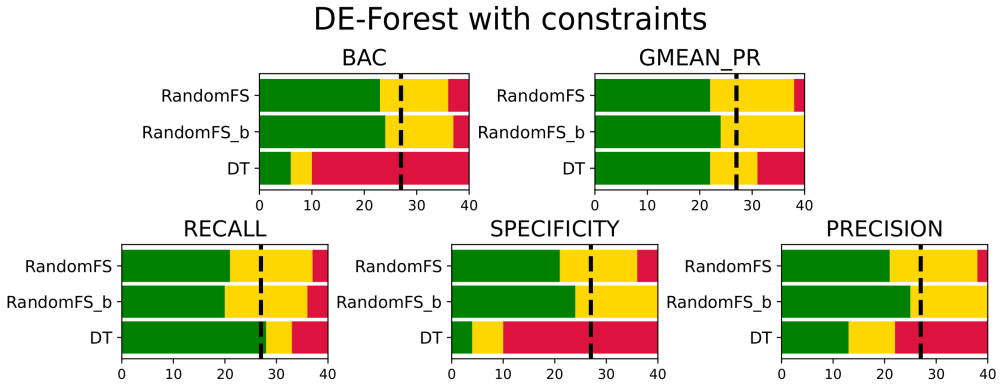


Figure 5.5: *The Wilcoxon test of DE-Forest method with constraints inside the optimization with reference methods (green – win, yellow – tie, red – loss)*

Bayesian Optimization showed the dependence of hyperparameter values and the impact of settings on the final quality of the model. Depending on the input data, the proposed method may take different parameters. When testing the method on imbalanced data, specific hyperparameters were selected.

RQ2: **How do various optimization criteria affect the *DE-Forest* method performance?**

DE optimization tested three criteria metrics. The fitness function (criteria) was one of the method's hyperparameters. The values were selected during experiment 1. The most common metric with the highest quality in experiment 1 was *BAC* and *AUC*. The final metric selection is *BAC*.

RQ3: **Can the proposed algorithm outperform state-of-the-art methods?**

The proposed algorithm achieves good results compared to the reference methods. *DE-Forest* outperforms with statistical significance with at least one reference method for each metric. When designing our method, we wanted an acceptable classification of imbalanced data, primarily a minority class, with which many classifiers do not cope well. The *DE-Forest* method obtains satisfactory results by analyzing the *Recall* metric reflecting the classification of the minority (positive) class.

The section proposed a novel classifier ensemble training method *DE-Forest*. Ensemble diversity was ensured by training each ensemble member on different attributes. Unlike algorithms known from the literature, such as *Random Forest*, the selection of the features was not random. Still, it resulted from an optimization algorithm based on *Differential Evolution* approach. Such an approach made it possible to build an ensemble of decision tree models based on regularly searched subsets of features, ensuring an appropriate diversity of the individual models. A vital element of the experiments was tuning the

hyperparameters of the *DE-Forest* using *Bayesian Optimization*, which allowed obtaining the best possible classifier for selected ranges of parameters and on selected imbalanced datasets. Such a procedure is essential if the method has to be applied to a real-life classification task. The experiments showed that the proposed approach achieves better results with statistical significance than reference methods such as DT classifier and the native *Random Forest* implementation. The promising results encourage us to continue working on developing the proposed method. Despite *DE-Forest* receiving good quality classifiers, it should be mentioned that the proposed metadata is characterized by high computational complexity and thus requires significantly more computational resources than reference.

5.2 MOOforest – multi-objective optimization to form decision tree ensemble

An essential element of forming ensembles is to keep a diverse pool of base classifiers so that the ensemble can leverage its strength. In the problem of imbalanced data, proper classification of minority data is particularly important, with which many non-ensemble classifiers cannot cope and direct their answers toward the majority class. One way to ensure this is feature selection, where different feature subspaces from the dataset are selected for each model. The primary method of randomly selecting attributes' subspaces is *Random Forest* [19]. The downside of this approach is randomness, which we do not influence, and it can lead to lower quality metrics and, thus, worse classification. The solution to this problem is the intelligent attribute selection done by single- or multi-objective optimization (MOO). Methods using one-criteria optimization only sometimes give satisfactory results. When objectives conflict, an aggregated metric often used as a single criterion does not offer the same capabilities as the MOO approach. In MOO, optimizing for many often incompatible criteria is possible, and a compromise is found. The solution is sought in the space of these criteria, and the MOO algorithm returns a set of solutions from which the user may choose the most compelling one. Increasingly, MOO optimization is used to train ensembles to classify imbalanced data.

The choice of the optimization algorithm is crucial to adequately searching the solution space. Wojciechowski [151] analyzed this space by comparing known optimization algorithms on many problems. The results are available in [2]. The author compared, among others, *NSGA-II* and *MOEA/D*. The results indicate that the *MOEA/D* algorithm produces solutions distributed more widely depending on the set reference directions vectors. The solutions are not concentrated in one place, allowing for the building of a diverse ensemble. Hence, it was decided to use *MOEA/D* in this research.

We propose the *MOOforest* [58] – classifier ensemble method using multi-objective optimization for imbalanced data classification. *MOOforest* employs multi-objective optimization, providing a diverse selection of feature subspaces for each DT model (base classifier). Figure 5.6 presents a schema of the training model procedure, and Algorithm 5.2 explains the method in more detail. These elements will allow us to describe how *MOOforest* works.

At first, the method gets learning set $\mathcal{LS}$. Then, such a set is transferred to the *MOEA/D* optimization algorithm with the appropriate parameters (step 2). To estimate classifier performance, $\mathcal{LS}$ is divided into training and test sets using cross-validation. The population $\hat{x}_s$ of sth generation is a vector of size $d \times n$, where d is the number of features for

[2] `https://github.com/swojciechowski/pymoo-benchmark`

a given dataset, and n is the set number of models in the ensemble. The initial population $\hat{x}_0$ is created during optimization algorithm initialization. The *MOEA/D* algorithm looks for different vector values based on the fitness function (eq. 5.1) and strives to maximize the *Precision* and *Recall* metrics simultaneously.

$$\begin{cases} \text{maximize } F_1(\hat{x}) = Precision \\ \text{maximize } F_2(\hat{x}) = Recall \end{cases} \tag{5.1}$$

The one feature vector V is later selected using the *PROMETHEE II* algorithm (step 3) and is transformed into a binary representation to indicate which feature was chosen unambiguously. Each real number in the vector is transformed binary representation, i.e., 1 means that the feature has been selected, 0 to the contrary. The vector with the solution V must be split into features for each model separately. Finally, the models are trained on the found subspaces (step 8) and added to the ensemble (step 9). The output of the algorithm is a classifier ensemble.

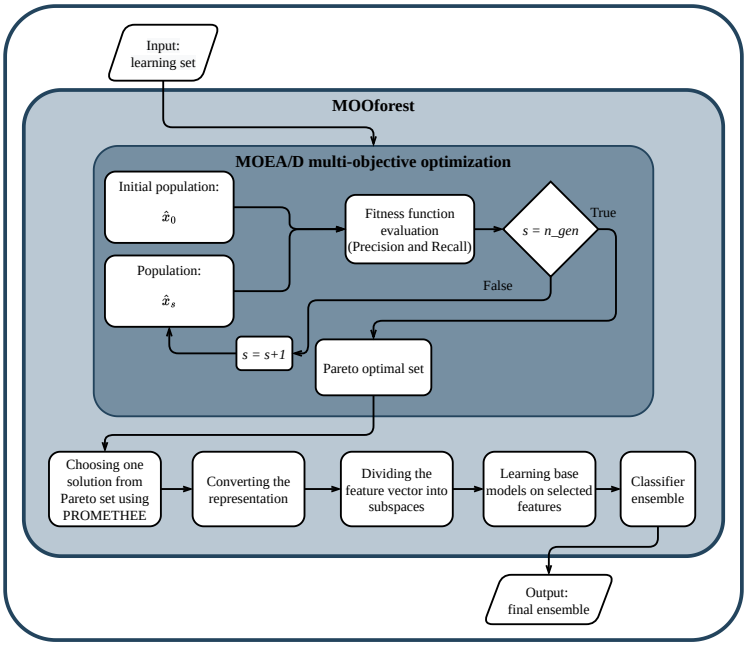


Figure 5.6: *Diagram of proposed method.*

Computational complexity analysis

The computational complexity of the proposed method consists of a few constituents. The complexity of the *CART Decision Tree* is $O(dN \times log_2 N)$, where d is the number of attributes, and N is the number of samples [124]. Thus, the computational complexity

Algorithm 5.2: *MOOforest method*

Input:
$\mathcal{LS} = \{(x_1, i_1), (x_2, i_2), \ldots, (x_N, i_N)\}$ – learning set
$TrainDT()$ – DT classifier training method based on the selected subset of features
PROMETHEE II – algorithm to choose one solution

Symbols:
n – number of models in the ensemble
d – number of features in the dataset

Output:
Π – pool of DT base classifiers

$\Pi \leftarrow \emptyset$
$PS \leftarrow$ Optimization($\mathcal{LS}$) ▷ PS is $(d * n)$ dimensional vector
$V \leftarrow PROMETHEE\ II(PS)$
for $j = 1$ to n **do**
  for $m = 1$ to d **do**
    $V[j + m - 2] \leftarrow \lceil V[j + m - 2] - 0.5 \rceil$
  end for
  $DT \leftarrow TrainDT([V[(j-1) * d + 1], ..., V[j * d]]$
  $\Pi \leftarrow \Pi \cup DT(V_j)$
end for

of training n decision trees is $O(ndN \times log_2 N)$. The main element of the computational complexity of the method is the *MOEA/D* optimization algorithm, whose computational complexity is $O(MUT)$ – M is the number of objectives, U is the population size, T is the size of the neighborhood [165]. In our case, $M = 2$, the complexity is $O(2UT)$. The entire computational complexity of the *MOOforest* method is $O(ndN \times log_2 N + MUT)$.

5.2.1 Experimental evaluation

The purpose of the experimental study is to answer the following research questions:

RQ1: **What are the non-dominated solutions characteristics returned by *MOOforest*?**

RQ2: **Does the number of optimization criteria significantly impact the final quality of the method?**

RQ3: **Can the *MOOforest* outperform state-of-the-art methods?**

Setup

The implementation of the experimental environment and methods are done using the *Python* programming language and a few libraries: *Pymoo* [14], *scikit-learn* [114], *Matplotlib* [67], *Numpy* [109], *Pandas* [149]. Complete source code, sufficient to repeat the experiments, is available at *GitHub repository*[3]. Additionally, we provided the complete results of the experiments that were conducted.

All used datasets are obtained from *Keel-dataset repository* [4], presented in Table 5.3 sorted by the *Imbalance Ratio*. Columns in this table are *ID* — the number of datasets, *Dataset* — the name of the dataset, *IR – Imbalance Ratio*, *Ex.* -– the number of instances, *Attr.* -– the number of attributes. These are binary problems.

Table 5.3: *Description of datasets*

ID	DATASET	IR	EX.	ATTR.	ID	DATASET	IR	EX.	ATTR.
1	*glass1*	1.82	214	9	19	*glass-0-1-6_vs_5*	19.44	184	9
2	*haberman*	2.78	306	3	20	*yeast-2_vs_8*	23.10	482	8
3	*yeast3*	8.10	1484	8	21	*flare-F*	23.79	1066	11
4	*page-blocks0*	8.79	5472	10	22	*yeast4*	28.10	1484	8
5	*yeast-2_vs_4*	9.08	514	8	23	*winequality-red-4*	29.17	1599	11
6	*ecoli-0-6-7_vs_3-5*	9.09	222	7	24	*poker-9_vs_7*	29.50	244	10
7	*glass-0-1-5_vs_2*	9.12	172	9	25	*winequality-white-9_vs_4*	32.60	168	11
8	*yeast-0-3-5-9_vs_7-8*	9.12	506	8	26	*abalone-17_vs_7-8-9-10*	39.31	2338	8
9	*yeast-0-2-5-6_vs_3-7-8-9*	9.14	1004	8	27	*abalone-21_vs_8*	40.50	581	8
10	*ecoli-0-1_vs_2-3-5*	9.17	244	7	28	*yeast6*	41.40	1484	8
11	*ecoli-0-2-6-7_vs_3-5*	9.18	224	7	29	*abalone-19_vs_10-11-12-13*	49.69	1622	8
12	*ecoli-0-6-7_vs_5*	10.00	220	6	30	*kr-vs-k-zero_vs_eight*	53.07	1460	6
13	*glass-0-1-6_vs_2*	10.29	192	9	31	*winequality-red-3_vs_5*	68.10	691	11
14	*ecoli-0-1-4-7_vs_2-3-5-6*	10.59	336	7	32	*abalone-20_vs_8-9-10*	72.69	1916	8
15	*glass-0-1-4-6_vs_2*	11.06	205	9	33	*kddcup-buffer_overflow_vs_back*	73.43	2233	41
16	*cleveland-0_vs_4*	12.31	173	13	34	*poker-8_vs_6*	85.88	1477	10
17	*page-blocks-1-3_vs_4*	15.86	472	10	35	*kddcup-rootkit-imap_vs_back*	100.14	2225	41
18	*abalone9-18*	16.40	731	8	36	*abalone19*	129.44	4174	8

The methods, their abbreviations, and the parameters used in the experiments:

- *MOOforest – Multi-Objective Optimization Forest* is the proposed method. The number of base models is 15 classifiers, the same for the other reference methods.
- *DT – CART Decision Tree Classifier* [20]. We used its *scikit-learn* implementation with default parameters such as:
  - *criterion of split – Gini impurity*

[3] `https://github.com/w4k2/MOOforest`

  - *maximum number of features* – *None* indicates no maximum value

- *DE-Forest* – *Differential Evolution Forest* ensemble with *DT* as the base classification model. It is our previous proposed method using single-objective optimization.

- *RandomFS* – based on the original article [19], *RandomFS* is a native implementation of *Random Forest*. The original concept relied on the majority vote. The value of the *RandomFS* parameter max features is $\sqrt{d}$, where d is the total number of features. Features are randomly picked for every model. *DT* as the base model has been selected using the same pre-presented default settings. Bootstrapping has not been used.

- *RandomFS_b* – the method described above (*RandomFS*) with the bootstrapping option set to True.

The reference methods selected for the experiments correspond to or are part of the proposed algorithm. *DT* is the base classifier used in *MOOforest*, so checking the *DT* performance can indicate whether creating an ensemble improves. The second reference method is *DE-Forest*, a single-criteria version of *MOOforest*. In this case, checking whether the multi-objective method can outperform the algorithm based on the aggregated criterion is essential. The last benchmark algorithm is *RandomFS* in two versions, i.e., *Random Forest* with or without bootstrapping. In this case, we inspect what wins: randomly selected features (*Random Forest*) or optimized feature selection according to two criteria (*MOOforest*).

MOOforest uses optimization algorithm *MOEA/D*, which is based on *Pymoo* implementation, with the following parameters: *number of generation* is 200, *number of neighbors* is 15, *sampling* is *Random*, *crossover* is *Simulated Binary*, *mutation* is *Polynomial*.

Methods compared during experiments based on stratified datasets using 5×2 *CV* [37]. We chose the widely used performance metrics to evaluate all methods: *Balanced Accuracy* (*BAC*), *Geometric Mean Score* (*Gmean*), *Recall*, *Specificity*, and *Precision*. *Gmean* is based on *Precision* and *Recall* metrics. To summarize the results for all datasets, the *Wilcoxon statistical rank-sum test* at a significance level of 0.05 was chosen [133].

5.2.2 Results

Since it is a multi-objective method, it returns a set of non-dominated solutions from which the user can choose the solution appropriate to the criteria he has set. Fig. 5.7 and 5.8 show an example set of solutions for selected datasets. Presenting graphs for all datasets is impossible because they are not average values. The solutions shown are in

Precision (x-axis) and *Recall* (y-axis) space. Blue points indicate different solutions. Fig. 5.7 shows the results for the *winequality-white* dataset. Even though, in this case, the method returns over 40 solutions, their values are located in three points. The solution set is not broad, unlike in the *yeast* dataset (Fig. 5.8). In this case, the algorithm found more solutions in a shape similar to the rotated exponential function. The user can choose the most essential solution because a larger pool of solutions is available. The number and arrangement of solutions in the fitness functions space depend primarily on the dataset and even on the selected part of the dataset after division using cross-validation. Metric values also vary due to data difficulties.

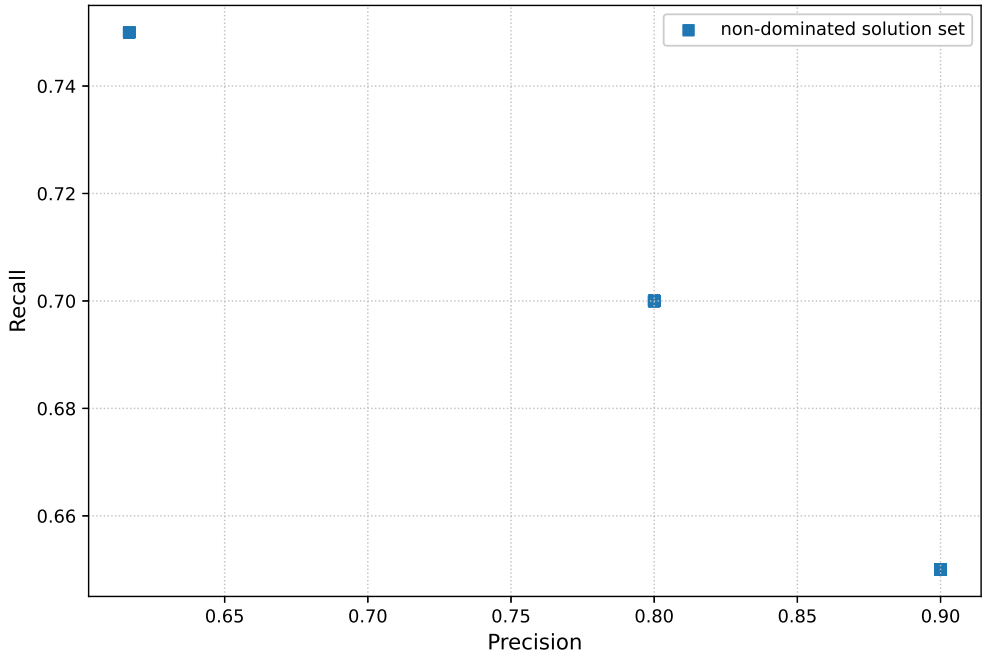


Figure 5.7: *Scatter plot of non-dominated solutions in Precision and Recall space – winequality-white-9_vs_4 dataset*

The final results averaged after cross-validation, and the standard deviation for the *Recall* (Tab. 5.4) and *Precision* (Tab. 5.5) metrics. The best result is in bold [4]. Only these metrics are shown because they are used as criteria inside optimization. Other results are included in the repository. Results of *MOOforest* are better than the reference methods for about half of the datasets. Optimization returns non-dominated solutions in the *Precision* and *Recall* space. The results presenting these solutions in figures are in the repository. Depending on the dataset and the selected fold, the charts may differ in the placement and number of non-dominated solutions. Some datasets give a broader non-dominated solution set, while others have solutions in only a few points.

[4]Detailed results are available in the GitHub repository [5]

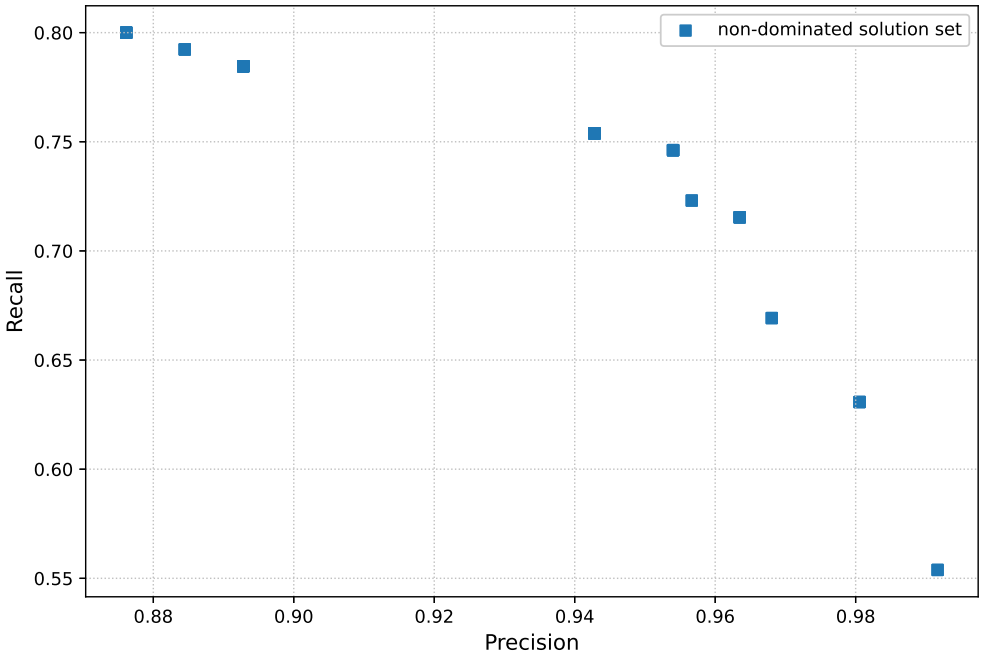


Figure 5.8: *Scatter plot of non-dominated solutions in Precision and Recall space – yeast-2_vs_4 dataset*

The summary results are presented in Fig. 5.9. The *Wilcoxon statistical rank-sum test* for five metrics compares the proposed *MOOforest* method with state-of-the-art methods on the left. The results have been averaged, and this ranking test shows wins in green, ties in yellow, and losses in red. If the dashed line on the graph is crossed in green, the *MOOforest* method won with statistical significance. The x-axis shows the number of tested datasets. The best results can be seen with the aggregate metric *Gmean*. The proposed method wins with statistical significance with four reference methods. *MOOforest* wins over the *RandomFS* and *RandomFS_b* methods, also above the level of statistical significance, for the remaining metrics (*BAC*, *Specificity*, *Precision*). On the other hand, the *Recall* metric shows a significant advantage of the proposed method over *DT*, which indicates a good classification of the minority class. The *MOOforest* method classifies imbalanced data well. Bootstrapping in the case of the tested data does not improve the classification quality at all, so this option was not used in the proposed method.

Based on Fig. 5.9, we can compare the impact of the number of criteria on the classification quality. The proposed *MOOforest* method uses multi-objective optimization and uses the basic *Precision* and *Recall* metrics as objectives. On the other hand, the *DE-Forest* method is a single-criteria version where the aim is the aggregated *Balanced Accuracy* metric. As previously described, *MOOforest* scores high on the *Gmean* metric, so compared to the *DE-Forest* method, it wins by statistical significance. It is better for

Table 5.4: *Results of Recall metric with the standard deviation, of all datasets sorted by IR. It is a comparison between proposed method and reference methods. The highest result for each metric and dataset is bold.*

ID	MOOforest	DT	DE-Forest	RandomFS	RandomFS-b
1	0.795 ± 0.032	0.731 ± 0.026	0.787 ± 0.031	**0.805 ± 0.046**	0.797 ± 0.039
2	0.706 ± 0.034	0.654 ± 0.027	0.673 ± 0.025	**0.724 ± 0.031**	0.724 ± 0.012
3	**0.939 ± 0.007**	0.932 ± 0.007	0.936 ± 0.008	0.890 ± 0.001	0.891 ± 0.001
4	**0.971 ± 0.003**	0.965 ± 0.003	0.971 ± 0.003	0.968 ± 0.003	0.967 ± 0.004
5	0.947 ± 0.008	0.941 ± 0.011	**0.950 ± 0.010**	0.916 ± 0.011	0.924 ± 0.017
6	**0.947 ± 0.018**	0.937 ± 0.021	0.941 ± 0.014	0.909 ± 0.012	0.904 ± 0.016
7	0.895 ± 0.019	0.853 ± 0.033	0.895 ± 0.013	0.893 ± 0.016	**0.897 ± 0.006**
8	**0.909 ± 0.004**	0.855 ± 0.019	0.906 ± 0.010	0.902 ± 0.001	0.902 ± 0.002
9	0.923 ± 0.005	0.896 ± 0.010	**0.927 ± 0.005**	0.907 ± 0.003	0.905 ± 0.003
10	**0.954 ± 0.012**	0.936 ± 0.012	0.948 ± 0.013	0.937 ± 0.012	0.934 ± 0.015
11	**0.938 ± 0.013**	0.931 ± 0.016	0.930 ± 0.017	0.904 ± 0.018	0.914 ± 0.013
12	**0.957 ± 0.015**	0.950 ± 0.021	0.950 ± 0.014	0.918 ± 0.014	0.917 ± 0.010
13	0.881 ± 0.027	0.851 ± 0.035	0.909 ± 0.007	0.903 ± 0.015	**0.914 ± 0.007**
14	0.951 ± 0.010	0.936 ± 0.020	**0.952 ± 0.009**	0.923 ± 0.007	0.927 ± 0.006
15	0.900 ± 0.021	0.847 ± 0.029	0.912 ± 0.009	0.912 ± 0.013	**0.918 ± 0.006**
16	0.926 ± 0.014	0.908 ± 0.026	**0.939 ± 0.015**	0.927 ± 0.009	0.930 ± 0.009
17	**0.995 ± 0.004**	0.984 ± 0.011	0.991 ± 0.007	0.987 ± 0.004	0.986 ± 0.005
18	**0.946 ± 0.007**	0.917 ± 0.009	0.946 ± 0.004	0.940 ± 0.006	0.943 ± 0.004
19	0.958 ± 0.021	0.964 ± 0.026	0.961 ± 0.018	**0.966 ± 0.011**	0.957 ± 0.010
20	**0.974 ± 0.005**	0.953 ± 0.011	0.969 ± 0.009	0.959 ± 0.001	0.959 ± 0.000
21	0.957 ± 0.002	0.939 ± 0.010	0.947 ± 0.008	**0.960 ± 0.001**	0.959 ± 0.001
22	0.963 ± 0.004	0.947 ± 0.006	**0.966 ± 0.003**	0.966 ± 0.001	0.966 ± 0.001
23	0.963 ± 0.002	0.937 ± 0.011	0.967 ± 0.001	0.967 ± 0.001	**0.967 ± 0.001**
24	**0.971 ± 0.010**	0.963 ± 0.012	0.967 ± 0.000	0.971 ± 0.004	0.967 ± 0.000
25	**0.974 ± 0.009**	0.952 ± 0.018	0.968 ± 0.008	0.971 ± 0.008	0.967 ± 0.010
26	0.974 ± 0.003	0.962 ± 0.005	**0.976 ± 0.001**	0.975 ± 0.001	0.975 ± 0.000
27	0.977 ± 0.008	0.971 ± 0.005	**0.977 ± 0.006**	0.977 ± 0.007	0.976 ± 0.003
28	0.976 ± 0.004	0.968 ± 0.007	**0.979 ± 0.003**	0.977 ± 0.001	0.976 ± 0.001
29	0.978 ± 0.003	0.960 ± 0.007	0.980 ± 0.001	0.980 ± 0.001	**0.980 ± 0.000**
30	0.995 ± 0.003	**0.996 ± 0.003**	0.995 ± 0.003	0.982 ± 0.001	0.982 ± 0.001
31	0.982 ± 0.005	0.973 ± 0.006	0.985 ± 0.002	0.985 ± 0.002	**0.986 ± 0.000**
32	0.985 ± 0.002	0.979 ± 0.003	0.986 ± 0.001	0.986 ± 0.000	**0.986 ± 0.000**
33	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**	0.999 ± 0.001
34	**0.989 ± 0.002**	0.980 ± 0.008	0.988 ± 0.001	0.988 ± 0.001	0.988 ± 0.001
35	1.000 ± 0.001	**1.000 ± 0.000**	0.999 ± 0.001	0.999 ± 0.001	1.000 ± 0.001
36	0.992 ± 0.000	0.983 ± 0.003	**0.992 ± 0.000**	0.992 ± 0.000	**0.992 ± 0.000**

almost all datasets tested. In the remaining metrics, *MOOforest* shows a better classification in about half of the datasets. The proposed method is worth using because it gives better results and has more possibilities.

5.2.3 Lesson learned

Based on the presented results, let us answer the research questions:

RQ1: **What are the non-dominated solutions characteristics returned by *MOOforest*?**

The Pareto front approximation is distinct depending on the data sample, i.e., the type of data – the selected dataset and the selected subset. For some problems,

Table 5.5: *Results of Precision metric with the standard deviation, of all datasets sorted by IR. It is a comparison between proposed method and reference methods. The highest result for each metric and dataset is bold.*

ID	MOOforest	DT	DE-Forest	RandomFS	RandomFS-b
1	0.794 ± 0.033	0.736 ± 0.028	0.788 ± 0.036	**0.803 ± 0.049**	0.799 ± 0.041
2	**0.670 ± 0.043**	0.659 ± 0.031	0.643 ± 0.031	0.621 ± 0.089	0.633 ± 0.083
3	**0.935 ± 0.008**	0.934 ± 0.008	0.931 ± 0.009	0.815 ± 0.044	0.815 ± 0.045
4	**0.970 ± 0.003**	0.965 ± 0.003	0.970 ± 0.003	0.967 ± 0.003	0.966 ± 0.004
5	0.945 ± 0.010	0.941 ± 0.010	**0.947 ± 0.012**	0.909 ± 0.035	0.916 ± 0.039
6	**0.946 ± 0.017**	0.939 ± 0.021	0.942 ± 0.013	0.873 ± 0.046	0.876 ± 0.046
7	0.844 ± 0.046	**0.852 ± 0.027**	0.836 ± 0.035	0.848 ± 0.045	0.812 ± 0.010
8	**0.897 ± 0.013**	0.862 ± 0.014	0.884 ± 0.035	0.828 ± 0.033	0.833 ± 0.041
9	0.913 ± 0.007	0.899 ± 0.009	**0.919 ± 0.007**	0.902 ± 0.014	0.883 ± 0.038
10	**0.953 ± 0.013**	0.938 ± 0.013	0.950 ± 0.012	0.939 ± 0.014	0.935 ± 0.022
11	0.935 ± 0.016	**0.936 ± 0.011**	0.922 ± 0.038	0.869 ± 0.050	0.886 ± 0.052
12	**0.957 ± 0.017**	0.951 ± 0.017	0.948 ± 0.018	0.891 ± 0.046	0.890 ± 0.044
13	0.845 ± 0.024	**0.861 ± 0.018**	0.841 ± 0.029	0.857 ± 0.038	0.855 ± 0.041
14	0.949 ± 0.009	0.942 ± 0.016	**0.951 ± 0.013**	0.912 ± 0.028	0.919 ± 0.028
15	0.863 ± 0.031	0.859 ± 0.019	0.845 ± 0.019	**0.864 ± 0.037**	0.859 ± 0.037
16	0.904 ± 0.034	0.906 ± 0.019	**0.919 ± 0.042**	0.886 ± 0.039	0.883 ± 0.040
17	**0.995 ± 0.004**	0.985 ± 0.012	0.991 ± 0.006	0.988 ± 0.004	0.985 ± 0.005
18	0.932 ± 0.013	0.919 ± 0.010	**0.933 ± 0.015**	0.914 ± 0.029	0.916 ± 0.026
19	0.961 ± 0.021	**0.970 ± 0.019**	0.948 ± 0.031	0.969 ± 0.008	0.939 ± 0.028
20	**0.974 ± 0.004**	0.956 ± 0.011	0.959 ± 0.023	0.923 ± 0.014	0.919 ± 0.000
21	0.933 ± 0.013	**0.935 ± 0.004**	0.932 ± 0.007	0.921 ± 0.002	0.921 ± 0.002
22	0.953 ± 0.008	0.949 ± 0.005	**0.954 ± 0.013**	0.932 ± 0.001	0.932 ± 0.001
23	0.937 ± 0.005	0.940 ± 0.005	**0.943 ± 0.014**	0.937 ± 0.006	0.938 ± 0.011
24	**0.963 ± 0.020**	0.957 ± 0.017	0.936 ± 0.000	0.956 ± 0.020	0.936 ± 0.000
25	**0.958 ± 0.022**	0.951 ± 0.016	0.946 ± 0.015	0.945 ± 0.018	0.941 ± 0.011
26	0.966 ± 0.004	0.963 ± 0.003	**0.970 ± 0.005**	0.956 ± 0.009	0.955 ± 0.008
27	**0.976 ± 0.008**	0.970 ± 0.006	0.967 ± 0.014	0.971 ± 0.012	0.960 ± 0.011
28	0.972 ± 0.005	0.973 ± 0.004	**0.976 ± 0.005**	0.956 ± 0.008	0.953 ± 0.001
29	0.962 ± 0.003	**0.962 ± 0.002**	0.961 ± 0.000	0.961 ± 0.000	0.961 ± 0.000
30	0.995 ± 0.004	**0.996 ± 0.003**	0.995 ± 0.003	0.963 ± 0.001	0.963 ± 0.001
31	0.972 ± 0.002	**0.973 ± 0.003**	0.971 ± 0.000	0.971 ± 0.000	0.971 ± 0.000
32	**0.979 ± 0.004**	0.978 ± 0.003	0.974 ± 0.002	0.973 ± 0.000	0.973 ± 0.000
33	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**	0.999 ± 0.001
34	0.979 ± 0.006	**0.979 ± 0.006**	0.977 ± 0.001	0.977 ± 0.001	0.977 ± 0.001
35	1.000 ± 0.001	**1.000 ± 0.000**	0.999 ± 0.001	0.999 ± 0.001	1.000 ± 0.001
36	0.985 ± 0.000	**0.985 ± 0.000**	0.985 ± 0.000	0.985 ± 0.000	0.985 ± 0.000

the optimization solution contains several dozen solutions distributed in the *Precision* and *Recall* space. Often, these solutions take the same *Precision* and *Recall*, marked as one point in the graphs, but the representation of these solutions may differ. In most datasets, non-dominated solutions have lower precision and recall metrics than the final classifier.

RQ2: **Does the number of optimization criteria significantly impact the final quality of the method?**

The number of criteria in the *MOOforest* and *DE-Forest* methods and the type of these criteria are of great importance for the final quality of the obtained classifier. As shown in the experimental part, the proposed method with multi-objective optimization achieves better results. One of the advantages of this method is the

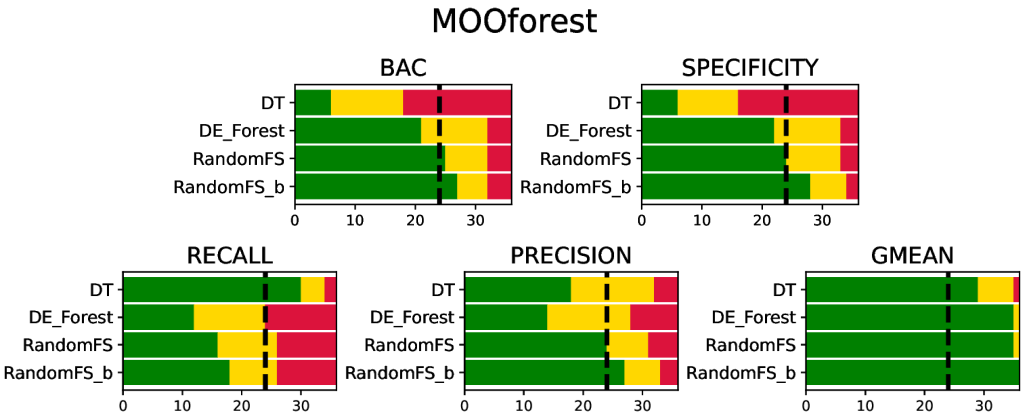


Figure 5.9: *The Wilcoxon test of MOOforest method with reference methods (green – win, yellow – tie, red – loss)*

selection of more specific and objective criteria than the aggregated metric. Multi-objective optimization returns a set of solutions from which we can choose a solution more suited to our needs and consider the criteria according to user selection.

RQ3: **Can the *MOOforest* outperform state-of-the-art methods?**

The *MOOforest* method has great potential compared to the reference methods. It wins for many datasets, which means it classifies the minority and majority classes well. In particular, it can be seen by analyzing the *Gmean* metric, in which the proposed method wins over most of the state-of-the-art algorithms. The advantage of the *MOOforest* method is that it is a set of solutions from which a potential user can choose a solution tailored to his/her needs. Depending on various criteria, the choice may fall on a solution directed more towards the first or second criterion, which allows the user to establish priorities.

The section presents a new *MOOforest* method for imbalanced data classification. It forms a classifier ensemble that builds models on feature subspaces obtained from multi-criteria optimization. Hence, the diversity of underlying models is ensured, and the subspaces are not randomized, as is the case in the classic *Random Forest* approach. The experimental analysis confirmed our assumptions about the benefits of using more criteria than just one aggregate optimization criterion, i.e., the returned solution based on the *Precision* and *Recall* criteria performed better than the solution based on the aggregated metric as *Balanced Accuracy*. Compared to other reference methods, the *MOOforest* proves to classify data well and gains an advantage in statistical tests. An interesting issue is the observation of the non-dominated solution set approximation. The non-dominated solutions may look different in each dataset and data sample's *Precision* and *Recall* spaces. The arrangement and number of solutions differ from the selected data slice.

Chapter 6

Multi Objective Local Optimization Forest

This chapter describes the *MOLO* method. It is a classifier ensemble built based on decision trees to classify imbalanced data. The local optimization algorithm adds different models to the ensemble, thus creating the ensemble that achieves the best classification quality at a given moment. Statistical analysis experiments compare the proposed method with state-of-the-art methods and allow answering research questions about the impact of the ensemble's size and local optimization' steps, the optimization criteria number, and the comparison of the proposed method with reference methods.

Multi-objective optimization offers much greater possibilities than single-objective optimization. By selecting appropriate criteria, the user can adjust the method and pay attention to critical elements. In ensemble formation, optimization can be used to create a subspace of features used to train each model, and the resulting ensemble is diverse. Additionally, the ensemble is not created randomly, as is the case with the popular *Random Forest* approach.

The proposed method *Multi-Objective Local Optimization Forest* (*MOLO*) is the ensemble composed of decision tree classifiers. The method creates many ensembles during local optimization by selecting base models in each iteration step. Ultimately, the final ensemble with the highest performance metric in imbalanced data classification is selected.

The operation of our method is presented in Alg. 6.1. Initially, the algorithm gets an $\mathcal{LS}$ dataset. It divides it into bootstraps $\{b_1, b_2, b_3, \ldots, b_B\}$, i.e., sets created by random drawing with replacement and creates a given number B of subsets (step 1.), as shown

in Fig. 6.1. Then, each such sample is used to build a DT model (step 8.). This way, we have an available pool of B models, from which we form the classifier ensemble.

Algorithm 6.1: *MOLO method*

Input:

$\mathcal{LS} = \{(x_1, i_1), (x_2, i_2), \ldots, (x_N, i_N)\}$ – learning set
B – maximum number of bootstrapped sets = maximum number of trained models
Ψ – model classifier
K_{steps} – maximum number of local optimization steps
P – the maximum number of ensembles in each level

Symbols:

$Train()$ – base classifier training method
b_i – ith bootstrapped sample
$prev$ – list of models from previous iteration
R – results from criteria metrics
$Evaluate$ – evaluate based on specified criteria
$FindNonDominated$ – find non-dominated solutions, Pareto front approximation
$final$ – final results from one optimization loop
ens – list of ensembles from all optimization steps
w – the worst ensembles in ens list according to one of criteria
ENSEMBLE – the final list of ensembles

Output:

Π – pool of DT base classifiers

1: $(b_i, b_B) \leftarrow$ resampling with replacement from $\mathcal{LS}$
2: $prev \leftarrow \emptyset$
3: $ens \leftarrow \emptyset$
4: **for** $k = 1$ to K_{steps} **do**
5: $\quad \Pi \leftarrow \emptyset$
6: $\quad \Pi \leftarrow \Pi \cup prev$
7: $\quad$ **for** $j = 1$ to m_B **do**
8: $\quad\quad \Psi_j \leftarrow Train(b)$
9: $\quad\quad \Pi \leftarrow \Pi \cup \Psi_j$
10: $\quad\quad R \leftarrow Evaluate(\Pi)$
11: $\quad$ **end for**
12: $\quad PF \leftarrow FindNonDominated(R)$
13: $\quad$ **if** $length(PF) \geq P$ **then**
14: $\quad\quad final \leftarrow ens - w$
15: $\quad$ **end if**
16: **end for**
17: $G \leftarrow$ the ensemble from $final$ with the highest performance metric
18: ENSEMBLE $\leftarrow G$

The main loop contains the most essential operations to obtain local optimization (step 4.). Fig. 6.2 shows an exemplary way of the ensemble forming. When $k = 1$, we have individual models trained on bootstraps, from which we create an ensemble in the next step. With each subsequent step, the number of models in the ensemble increases, i.e.,

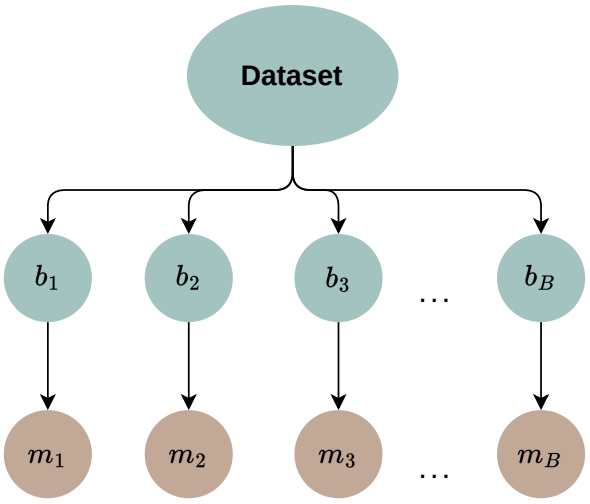


Figure 6.1: *Bootstrapping and learning base models*

$k = 2$ – two models, $k = 3$ – three models, etc. It is a form of pruning described in Sect. 2.2 based on rankings from the selected metrics, also known as order-based pruning.

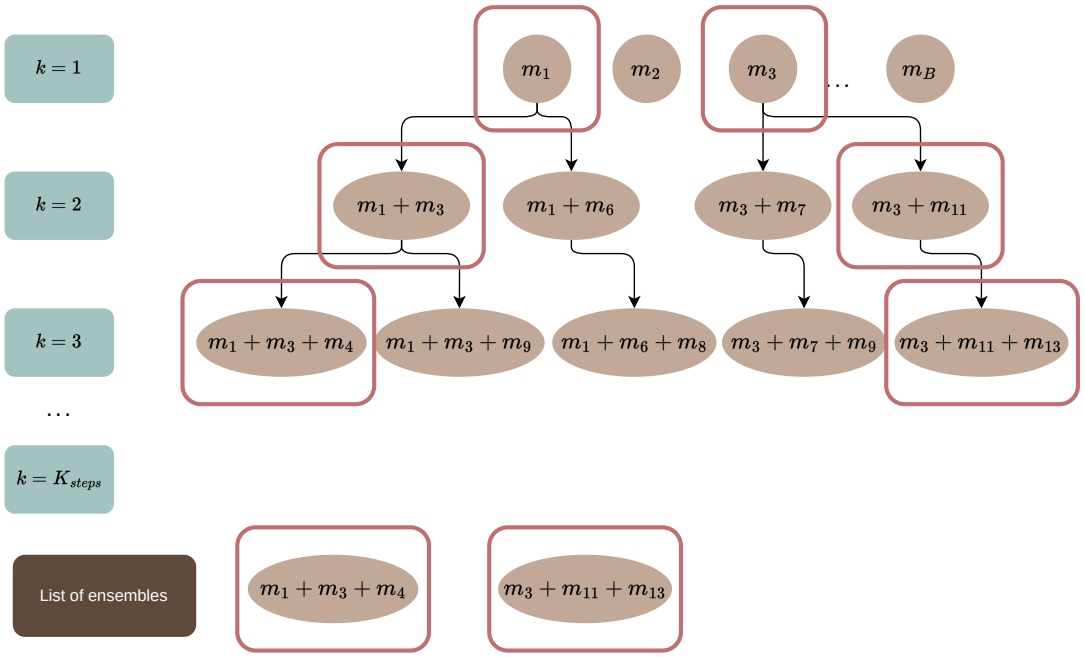


Figure 6.2: *Optimization steps, creating ensembles and pruning*

Let us follow the algorithm process for $k = 2$ in more detail. All possible pairs of models from the pool $[m_1, \ldots, m_B]$ are created. In the case of $k = 3$ and more, models from the previous iteration of the selected ensembles should be added to the created ensemble

(step 6.), and then each model from the pool $[m_1, m_B]$ (step 9.). Then, the metrics (criteria) are calculated for each ensemble (step 10.) – the *Evaluate* function. The function is based on two

$$\max_{x \in X}(f_1(x), f_2(x))$$

or three

$$\max_{x \in X}(f_1(x), f_2(x), f_3(x))$$

criteria. In the case of two criteria, we have two different sets. The first contains *Hellinger Distance* and the *Gmean* quality metric 6.1, and the second – *Margin* and the *Diversity* metric 6.2. The three-criteria case is a specific combination of non-aggregated metrics, and they are: *Margin*, *Diversity* and *Hellinger Distance* 6.3.

$$\begin{cases} \text{maximize } F_1(\hat{x}) = HellingerDistance \\ \text{maximize } F_2(\hat{x}) = Gmean \end{cases} \tag{6.1}$$

$$\begin{cases} \text{maximize } F_1(\hat{x}) = Margin \\ \text{maximize } F_2(\hat{x}) = Diversity \end{cases} \tag{6.2}$$

$$\begin{cases} \text{maximize } F_1(\hat{x}) = Margin \\ \text{maximize } F_2(\hat{x}) = Diversity \\ \text{maximize } F_3(\hat{x}) = HellingerDistance \end{cases} \tag{6.3}$$

Margin [104] is defined as follows:

$$f_m(x_i) = log\left(\left|\frac{v_{y_i}^{(i)} - v_{\hat{y}_i}^{(i)}}{\mathcal{M}}\right|\right) \tag{6.4}$$

where $\mathcal{M}$ – number of classes, x_i – attribute vector values, y_i – the true class label, v – the number of votes for specific class; and *Diversity* [104]:

$$f_d(x_i) = log(\frac{v_{y_i}^{(i)}}{\mathcal{M}}) \tag{6.5}$$

The *Hellinger Distance* is a similarity between the probabilities P_1 and P_2. It uses the *True Positive Rate* (TPR) and the *False Positive Rate* (FPR), and it is formulated as follows [31]:

$$d_H(TPR, FPR) = \sqrt{(\sqrt{TPR} - \sqrt{FPR})^2 + (\sqrt{1 - TPR} - \sqrt{1 - FPR})^2} \tag{6.6}$$

Since we have two or three criteria, it is not possible to select the best classifiers unequivocally, and it is required to find non-dominated solutions. The $FindNonDominated$ function (Alg. 6.2) searches the list with values from the criteria and rejects all values less than other points in this list. The non-dominated solution set is created, which takes into account two or more criteria at the same time.

Algorithm 6.2: *FindNonDominated function*

Input:
R – results from criteria metrics

Output:
PF – the list of non-dominated solutions

```
1: PF ← ∅
2: M_max ← length(R)
3: for m = 1 to M_max do
4:     if R_m ≥ R_{m+1} then
5:         PF ← PF ∪ R_m
6:     end if
7: end for
```

The last element in the main loop is the pruning of the ensemble, which uses *Beam search*. It makes the method faster because it does not store all the ensembles in memory, and with large numbers of B and K_{steps}, this could prove to be a challenge. The parameter P set at the beginning determines the maximum number of ensembles saved in each iteration (steps 13., 14.). Fig. 6.2 shows an example. Selected models are marked with a red frame. In step $k = 1$, m_1, and m_3 are selected, then in step $k = 2$, each combination with these selected models is checked, and the metrics are calculated to select the best ensemble. The models m_3 and m_{11} were selected in the example. This procedure is repeated as often as K_{steps}. Finally, we have two ensembles $[m_1, m_3, m_4]$ and $[m_3, m_{11}, m_{13}]$. Having a list of ensembles, we can choose the final classifier with the highest performance metric (steps 17., 18.).

Computational complexity analysis

The fundamental element of computational complexity is the proposed approach's complexity. It is $O(P)$, where P is the maximum number of ensembles saved in each iteration. The method uses *CART Decision Tree* whose complexity is $O(dN \times log_2 N)$, where d is the number of attributes, and N is the number of samples [124]. Suppose we train n decision trees, then the complexity is $O(ndN \times log_2 N)$. The total computational complexity of the proposed *MOLO* method is $O(P + ndN \times log_2 N)$.

6.1 Experimental evaluation

To confirm the suggested strategy's efficacy, we ran several computer tests. We aimed to address the following research questions during them:

RQ1: **What effect does the number of candidate models have on the quality of the method?**

RQ2: **Does the number of criteria affect the quality of the *MOLO* method positively or negatively?**

RQ3: **Can the suggested approach outperform state-of-the-art methods?**

6.1.1 Setup

This section comprehensively describes all the elements required to carry out valid experiments. The description makes it feasible to replicate the study.

All experiments were performed in the *Python* programming language using several libraries: *scikit-learn* [114], *Imbalanced-learn* [86], *Numpy* [109], *Matplotlib* [67], *Pandas* [149]. Detailed results of the experiments are available on the *GitHub repository*[1].

The methods are tested on many imbalanced datasets, which are considered benchmark datasets [107]. The selected datasets are a two-class problem with a wide range of class imbalances. The datasets were fetched from the *Keel-dataset repository*[4]. Columns in the Table 6.1 are as follows: *ID* – the number of the dataset, *Dataset* – the name of the

[1] `https://github.com/w4k2/MOLO-Forest`

dataset, *IR* – the Imbalance Ratio, *Ex.* – the number of instances, *Attr.* – the number of attributes.

Table 6.1: *Description of datasets*

ID	DATASET	IR	EX.	ATTR.	ID	DATASET	IR	EX.	ATTR.
1	*glass1*	1.82	214	9	31	*zoo-3*	19.20	101	16
2	*pima*	1.87	768	8	32	*glass-0-1-6_vs_5*	19.44	184	9
3	*glass0*	2.06	214	9	33	*yeast-1-4-5-8_vs_7*	22.10	693	8
4	*yeast1*	2.46	1484	8	34	*glass5*	22.78	214	9
5	*haberman*	2.78	306	3	35	*yeast-2_vs_8*	23.10	482	8
6	*vehicle1*	2.90	846	18	36	*flare-F*	23.79	1066	11
7	*vehicle3*	2.99	846	18	37	*yeast4*	28.10	1484	8
8	*ecoli1*	3.36	336	7	38	*winequality-red-4*	29.17	1599	11
9	*ecoli2*	5.46	336	7	39	*poker-9_vs_7*	29.50	244	10
10	*yeast3*	8.10	1484	8	40	*yeast-1-2-8-9_vs_7*	30.57	947	8
11	*ecoli3*	8.60	336	7	41	*winequality-white-9_vs_4*	32.60	168	11
12	*page-blocks0*	8.79	5472	10	42	*yeast5*	32.73	1484	8
13	*yeast-2_vs_4*	9.08	514	8	43	*winequality-red-8_vs_6*	35.44	656	11
14	*ecoli-0-6-7_vs_3-5*	9.09	222	7	44	*ecoli-0-1-3-7_vs_2-6*	39.14	281	7
15	*glass-0-1-5_vs_2*	9.12	172	9	45	*abalone-17_vs_7-8-9-10*	39.31	2338	8
16	*yeast-0-3-5-9_vs_7-8*	9.12	506	8	46	*abalone-21_vs_8*	40.50	581	8
17	*yeast-0-2-5-6_vs_3-7-8-9*	9.14	1004	8	47	*yeast6*	41.40	1484	8
18	*ecoli-0-1_vs_2-3-5*	9.17	244	7	48	*winequality-white-3_vs_7*	44.00	900	11
19	*ecoli-0-2-6-7_vs_3-5*	9.18	224	7	49	*winequality-red-8_vs_6-7*	46.50	855	11
20	*yeast-0-5-6-7-9_vs_4*	9.35	528	8	50	*abalone-19_vs_10-11-12-13*	49.69	1622	8
21	*ecoli-0-6-7_vs_5*	10.00	220	6	51	*kr-vs-k-zero_vs_eight*	53.07	1460	6
22	*glass-0-1-6_vs_2*	10.29	192	9	52	*winequality-white-3-9_vs_5*	58.28	1482	11
23	*ecoli-0-1-4-7_vs_2-3-5-6*	10.59	336	7	53	*poker-8-9_vs_6*	58.40	1485	10
24	*glass-0-1-4-6_vs_2*	11.06	205	9	54	*winequality-red-3_vs_5*	68.10	691	11
25	*glass2*	11.59	214	9	55	*abalone-20_vs_8-9-10*	72.69	1916	8
26	*cleveland-0_vs_4*	12.31	173	13	56	*kddcup-buffer_overflow_vs_back*	73.43	2233	41
27	*yeast-1_vs_7*	14.30	459	7	57	*poker-8-9_vs_5*	82.00	2075	10
28	*glass4*	15.46	214	9	58	*poker-8_vs_6*	85.88	1477	10
29	*page-blocks-1-3_vs_4*	15.86	472	10	59	*kddcup-rootkit-imap_vs_back*	100.14	2225	41
30	*abalone9-18*	16.40	731	8	60	*abalone19*	129.44	4174	8

The experiments compare six of our methods proposed in the article with seven reference methods. Table 6.2 lists all tested methods and highlights the criteria within those methods. Legend of markings in methods: *H* – *Hellinger*, *G* – *Gmean*, *M* – *Margin*, *D* – *Diversity*, *RE* – *Reduced Error*, *C* – *Complementariness*, *m* – *Margin* (the implementation from library), *h* – holdout.

All methods use a *DT* classifier as the base model and *Majority Voting* as a prediction. *MOLO* methods assume values of the number of bootstrap parameter from 25 to 200. The number of bootstrap means the number of candidate models to enter the ensemble. Optimization within these methods is performed by $K_{steps} = 20$ iterations. Pruning value $P = 10$, so there can be a maximum of ten ensembles after each optimization

Method / Criterion	*Hellinger*	*Gmean*	*Margin*	*Diversity*	*Reduced Error*	*Complementariness*	None
MOLO-HG	✓	✓					
MOLO-MD			✓	✓			
MOLO-MDH	✓		✓	✓			
MOLO-HG_h	✓	✓					
MOLO-MD_h			✓	✓			
MOLO-MDH_h	✓		✓	✓			
Ens-all							✓
OBPE-M			✓				
OBPE-D				✓			
OBPE-MD			✓	✓			
OBPE-RE					✓		
OBPE-C						✓	
OBPE-m			✓				

Table 6.2: *Criteria for each of the tested methods*

step. *MOLO* methods ending in h (holdout) mean that inside the method, there is an additional division of data into a training part (60%) and a test part (30%). Bootstraps are determined from the training part. Methods without this option are used as a test to set all method inputs (X, y) and bootstrap them.

The *Ens-all* method is an ensemble comprising all candidate base models trained on bootstrap (100 bootstraps). The *OBPE* (Order Base Pruning Ensemble) methods [104] are reference methods in a version with different criteria marked in Table 6.2. These methods have the following parameters: the number of bootstraps equals 100, and the pruning value equals 20. In the case of the *OBPE-MD* method containing aggregated *Margin* and *Diversity* criteria, the value $\alpha = 0.5$. Other parameters not listed have default values.

All these methods are compared in experiments on sixty datasets using 5×2 *CV* [37].

We chose the broadly utilized measurements to assess all methods: *Balanced Accuracy* (*BAC*), *Geometric Mean Score* (*Gmean*), *Recall*, *Specificity* and *Precision*. *Gmean* is based on *Precision* and *Recall* metrics.

Statistical tests are needed to summarize the results for all datasets; in this case, the *Wilcoxon statistical rank-sum test* at a significance level of 0.05 [133] is used.

6.2 Results

Two experiments were designed to answer the research questions. The first examines the influence of the number of models in the ensemble, while the second compares the proposed methods with state-of-the-art methods. The experiments carried out will help answer the previously posed research questions.

6.2.1 Experiment 1

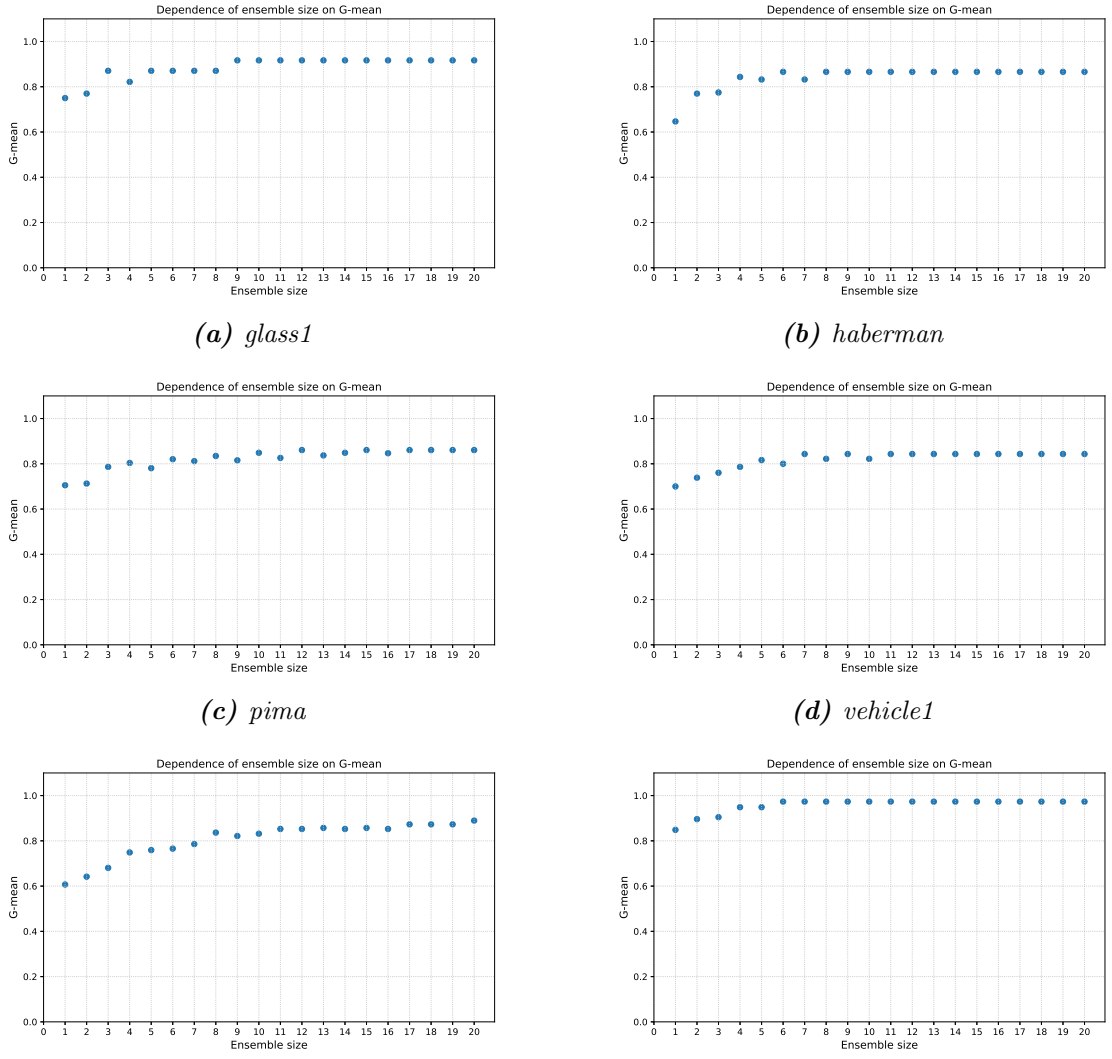
One of the parameters of the *MOLO* methods is the ensemble size, i.e., the number of base models in the classifier ensemble. At the same time, it is the K_{steps} parameter defining the maximum number of steps in the local optimization. Therefore, this experiment investigates the dependence of ensemble size on *Gmean* quality and, simultaneously, allows for determining the value of the K_{steps} parameter.

The results for the selected datasets are presented in Fig. 6.3. The y-axis indicates the number of models in the ensemble, ranging from 1 to 20, and the x-axis is the *Gmean* metric, a criterion measured within the method, where the value of 1 means the best quality. Depending on the problem, the value of ensemble size varies. For most datasets, *Gmean* is set to a given level after reaching ensemble size values above 5 and below 10 (Fig. 6.3a, 6.3b, 6.3f). In Fig. 6.3c, the value stabilizes, but quite late, around model 17. In the case of the *vehicle3* dataset (Fig. 6.3e), the *Gmean* metric continues to increase as the ensemble grows.

It is impossible to determine one best value of the K_{steps} parameter for so many datasets, so ultimately, the parameter $K_{steps} = 20$ for further experiments. This value allows for obtaining an outstanding quality *Gmean* metric for most datasets and thus does not cause excessive calculations. The larger K_{steps} was, the more loops would have to be performed in the optimization, which would significantly extend the computation time and would not provide measurable results. Suppose there is one problem, i.e., a specific dataset in which classification is to be performed. In that case, this parameter can be adapted to the needs of such a problem, ensuring maximum quality and reduction of calculations.

6.2.2 Experiment 2

The main experiment is to compare the proposed approach with other reference methods using the statistical test. The *Wilcoxon statistical rank-sum test* checks whether the selected method is statistically significantly better than the other in pairwise rankings.



(a) *glass1* ***(b)*** *haberman*

(c) *pima* ***(d)*** *vehicle1*

(e) *vehicle3* ***(f)*** *yeast3*

Figure 6.3: *Dependence of the number of models in the classifier ensemble on the Gmean metric*

After preliminary analysis, the *MOLO_MDH* method was selected for comparison with other methods. There are three colors in Fig. 6.4 and they indicate respectively: green – the *MOLO_MDH* method won compared to another method (y-axis) on the number of datasets given on the x-axis, yellow – it tied, red – it lost. The black dashed line is an indication of statistical significance. If the number of wins (green) exceeds the dashed line at approximately 38 datasets, the *MOLO_MDH* method wins with statistical significance over the compared method. The rankings were performed for four metrics: *BAC*, *Recall*, *Precision*, *Gmean*.

When comparing the 3-criteria and 2-criteria *MOLO* method, *MOLO_MDH* wins for a dozen or so datasets, loses for a similar number of datasets (depending on the metric), but usually there is a draw for about 30 datasets. *MOLO* methods are comparable, and

there is no advantage of any of them.

The best results comparing *MOLO_MDH* with reference methods are shown for the *Recall* metric. The proposed method wins with statistical significance and outperforms the following methods based on the *Margin*, *Diversity*, *Reduced Error*, and *Complementariness* criteria (*OBPE-M*, *OBPE-D*, *OBPE-MD*, *OBPE-m*, *OBPE-RE*, and *OBPE-C*), and *Ens-all*. High scores for the *Recall* metric reflect the good classification of the minority class, which is important in classifying imbalanced data. The method shows the least advantage for the *Precision* metric. *Gmean* is a metric averaged from *Precision* and *Recall*, so despite high *Recall*, *Gmean* does not achieve statistical significance over state-of-the-art methods. The *BAC* show statistical significance for *MOLO_MDH* over four methods *OBPE-M*, *OBPE-D*, *OBPE-MD*, *OBPE-m*.

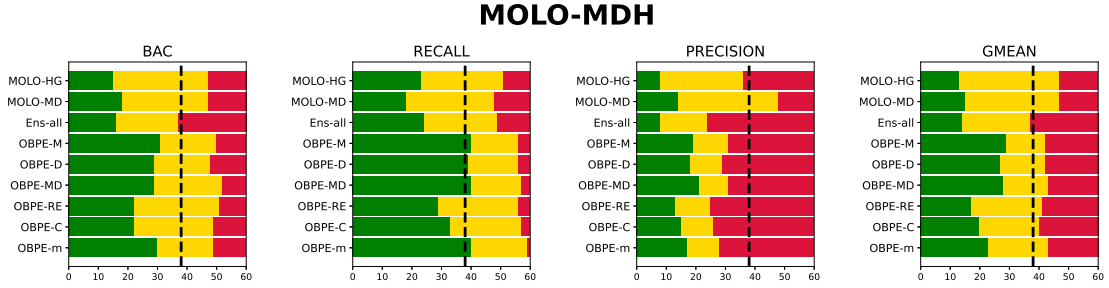


Figure 6.4: *Wilcoxon ranking, comparison with reference methods*

6.3 Lesson learned

Extensive research allows us to answer the following questions.

RQ1: **What effect does the number of candidate models have on the quality of the method?**

The number of models is one of the parameters of the proposed *MOLO* method, which reflects not only the size of the ensemble but also the number of steps in local optimization (parameter K_{steps}). Generally, the larger the size of the classifier ensemble, the better the classification and, therefore, the higher the performance metric value. However, in some cases, there is a point at which this relationship no longer holds, and increasing the number of models does not change the classification quality. It all depends on the problem, i.e., the dataset on which the classification takes place. 60 datasets were examined in the experiments, and the value of the K_{steps} parameter was set to 20 to ensure stabilization of the best possible quality for datasets that need more models in the ensemble for this purpose.

RQ2: **Does the number of criteria affect the quality of the *MOLO* method positively or negatively?**

The criteria adopted by the *MOLO* method are *Margin*, *Diversity*, and *Hellinger Distance*. The number of criteria in the *MOLO* method does not impact quality much as one might expect. None of the methods shows a statistically significant difference between the *MOLO* methods.

RQ3: **Can the suggested approach outperform state-of-the-art methods?**

Yes, especially for the *Recall* metric, which translates into a good classification of the majority class, vital in the case of imbalanced data. Average metrics (*BAC* and *Gmean*) also confirm some advantages over the other methods.

The chapter proposes a novel *MOLO* method for building ensembles employing local optimization using bootstrapping and order-based pruning. The ensemble consists of decision tree models and classifies imbalanced data. *MOLO* can accept two or three optimization criteria, including performance metrics and diversity. The conducted research shows the advantage of the proposed method over state-of-the-art methods.

Chapter 7

Comparison of proposed algorithms

The chapter summarizes performance of all the methods proposed in this work. We compared the methods on selected datasets through a statistical test and tables with detailed results for individual metrics. A thorough analysis of the results allowed for describing the purpose of each proposed method.

Since almost all the methods proposed in this work are ensemble classifiers, it is possible to compare them. The exception is the method proposed in Chapter 3 – an algorithm for feature selection. This algorithm prepares data, especially with a cost assigned to each feature. So, in this particular case, the algorithm is not a classifier. Well-known classifiers can use it to select features. In the case of our four proposed classifiers, feature selection is built-in, and there is no need to use it a second time in the data preparation phase.

Each of the methods compared in the experiments is described in detail in individual chapters:

- *SEMOOS – SVM Ensemble with Multi-Objective Optimization Selection* (Section 4)
- *DE-Forest – Differential Evolution Forest* (Section 5.1)
- *MOOforest – Multi-Objective Optimization Forest* (Section 5.2
- *MOLO – Multi Objective Local Optimization Forest* (Section 6)

All methods take their default parameters. The research is carried out on selected datasets from the *Keel repository*, sorted by increasing *Imbalance Ratio* (*IR*). The experimental protocol is 5×2 *CV*. The results are for the following metrics: *BAC*, *Gmean*,

Recall, *Specificity*, and *Precision*. The tables show the average results for each dataset separately, while the graphs are the *Wilcoxon rank-sum test* at a significance level of 0.05.

The tables present average results from folds and standard deviation for selected datasets and the four compared methods. Depending on the metric, different methods show advantages. The best result for each dataset is bolded, thus ensuring better readability of the table.

For the *Balanced Accuracy* metric (Tab. 7.1), the *MOLO-MDH* method achieves the highest values. It has lower values for a few datasets, but the difference is insignificant and usually within the standard deviation.

Table 7.1: *Balanced Accuracy*

DATASET	**MOOforest**	**SEMOOS**	**DE-Forest**	**MOLO-MDH**
glass1	**0.758 ± 0.038**	0.501 ± 0.028	0.740 ± 0.033	0.736 ± 0.033
haberman	0.557 ± 0.037	0.513 ± 0.017	0.537 ± 0.031	**0.568 ± 0.042**
yeast3	0.806 ± 0.025	0.813 ± 0.031	0.759 ± 0.048	**0.843 ± 0.026**
page-blocks0	0.905 ± 0.016	0.771 ± 0.036	0.902 ± 0.014	**0.914 ± 0.011**
yeast-2-vs-4	0.788 ± 0.047	0.758 ± 0.053	0.801 ± 0.042	**0.839 ± 0.058**
ecoli-0-6-7-vs-3-5	0.813 ± 0.055	0.667 ± 0.088	0.749 ± 0.076	**0.826 ± 0.069**
yeast-0-3-5-9-vs-7-8	0.578 ± 0.022	0.566 ± 0.025	0.560 ± 0.034	**0.628 ± 0.033**
yeast-0-2-5-6-vs-3-7-8-9	0.681 ± 0.021	0.614 ± 0.037	0.680 ± 0.028	**0.730 ± 0.028**
ecoli-0-1-vs-2-3-5	**0.796 ± 0.051**	0.696 ± 0.074	0.748 ± 0.070	0.793 ± 0.062
ecoli-0-2-6-7-vs-3-5	0.763 ± 0.079	0.661 ± 0.079	0.686 ± 0.095	**0.826 ± 0.043**
ecoli-0-6-7-vs-5	0.828 ± 0.094	0.745 ± 0.093	0.784 ± 0.092	**0.864 ± 0.060**
ecoli-0-1-4-7-vs-2-3-5-6	0.780 ± 0.054	0.703 ± 0.090	0.751 ± 0.041	**0.817 ± 0.074**
cleveland-0-vs-4	0.616 ± 0.081	0.693 ± 0.112	0.620 ± 0.092	**0.703 ± 0.084**
page-blocks-1-3-vs-4	**0.964 ± 0.028**	0.694 ± 0.077	0.938 ± 0.039	0.930 ± 0.052
abalone9-18	0.602 ± 0.046	**0.671 ± 0.054**	0.560 ± 0.025	0.632 ± 0.046
glass-0-1-6-vs-5	**0.786 ± 0.131**	0.638 ± 0.121	0.676 ± 0.153	0.767 ± 0.153
yeast-2-vs-8	**0.728 ± 0.080**	0.725 ± 0.034	0.673 ± 0.124	0.727 ± 0.071
yeast4	0.568 ± 0.028	0.531 ± 0.015	0.546 ± 0.029	**0.600 ± 0.036**
poker-9-vs-7	**0.623 ± 0.097**	0.500 ± 0.000	0.500 ± 0.000	0.570 ± 0.095
abalone-17-vs-7-8-9-10	0.570 ± 0.030	0.587 ± 0.048	0.558 ± 0.027	**0.620 ± 0.026**
abalone-21-vs-8	0.710 ± 0.084	0.678 ± 0.097	0.591 ± 0.096	**0.738 ± 0.091**
yeast6	0.618 ± 0.044	0.540 ± 0.041	0.602 ± 0.048	**0.660 ± 0.036**
kr-vs-k-zero-vs-eight	0.887 ± 0.068	0.622 ± 0.085	0.887 ± 0.082	**0.972 ± 0.055**
abalone-20-vs-8-9-10	0.553 ± 0.043	0.557 ± 0.060	0.504 ± 0.011	**0.590 ± 0.039**
kddcup-buffer-overflow-vs-back	**1.000 ± 0.000**	0.983 ± 0.017	**1.000 ± 0.000**	**1.000 ± 0.000**
poker-8-vs-6	0.531 ± 0.094	**0.603 ± 0.173**	0.500 ± 0.000	0.542 ± 0.080
kddcup-rootkit-imap-vs-back	0.982 ± 0.030	0.982 ± 0.022	0.959 ± 0.032	**1.000 ± 0.000**

In the case of the *Gmean* metric (Tab. 7.2), *MOOforest* wins, achieving the best results for all datasets even though the *Gmean* metric is an aggregated metric from *Precision*

(Tab. 7.3) and *Recall* (Tab. 7.4), for these two metrics, *MOOforest* does not have the highest scores for each dataset but still has the majority. *DE-Forest* best classified the remaining datasets.

Table 7.2: *Gmean*

DATASET	**MOOforest**	**SEMOOS**	**DE-Forest**	**MOLO-MDH**
glass1	**0.795 ± 0.032**	0.151 ± 0.157	0.738 ± 0.034	0.659 ± 0.042
haberman	**0.688 ± 0.038**	0.155 ± 0.144	0.519 ± 0.035	0.358 ± 0.070
yeast3	**0.937 ± 0.007**	0.794 ± 0.039	0.737 ± 0.060	0.732 ± 0.030
page-blocks0	**0.971 ± 0.003**	0.736 ± 0.051	0.899 ± 0.015	0.858 ± 0.013
yeast-2-vs-4	**0.946 ± 0.009**	0.717 ± 0.080	0.786 ± 0.050	0.711 ± 0.081
ecoli-0-6-7-vs-3-5	**0.946 ± 0.018**	0.542 ± 0.215	0.718 ± 0.098	0.688 ± 0.117
yeast-0-3-5-9-vs-7-8	**0.903 ± 0.008**	0.360 ± 0.077	0.435 ± 0.066	0.340 ± 0.053
yeast-0-2-5-6-vs-3-7-8-9	**0.918 ± 0.006**	0.474 ± 0.087	0.632 ± 0.040	0.552 ± 0.049
ecoli-0-1-vs-2-3-5	**0.954 ± 0.012**	0.618 ± 0.122	0.717 ± 0.087	0.656 ± 0.092
ecoli-0-2-6-7-vs-3-5	**0.936 ± 0.014**	0.532 ± 0.202	0.625 ± 0.152	0.682 ± 0.059
ecoli-0-6-7-vs-5	**0.957 ± 0.016**	0.683 ± 0.153	0.758 ± 0.119	0.741 ± 0.058
ecoli-0-1-4-7-vs-2-3-5-6	**0.950 ± 0.010**	0.621 ± 0.152	0.722 ± 0.053	0.646 ± 0.128
cleveland-0-vs-4	**0.915 ± 0.023**	0.602 ± 0.175	0.504 ± 0.175	0.470 ± 0.136
page-blocks-1-3-vs-4	**0.995 ± 0.004**	0.610 ± 0.138	0.936 ± 0.041	0.850 ± 0.077
abalone9-18	**0.939 ± 0.010**	0.580 ± 0.094	0.402 ± 0.056	0.366 ± 0.114
glass-0-1-6-vs-5	**0.959 ± 0.021**	0.433 ± 0.307	0.556 ± 0.264	0.544 ± 0.255
yeast-2-vs-8	**0.974 ± 0.005**	0.669 ± 0.048	0.561 ± 0.228	0.506 ± 0.101
yeast4	**0.958 ± 0.006**	0.234 ± 0.089	0.336 ± 0.091	0.281 ± 0.089
poker-9-vs-7	**0.967 ± 0.014**	0.000 ± 0.000	0.178 ± 0.000	0.183 ± 0.247
abalone-17-vs-7-8-9-10	**0.970 ± 0.003**	0.388 ± 0.157	0.363 ± 0.066	0.308 ± 0.050
abalone-21-vs-8	**0.977 ± 0.008**	0.552 ± 0.228	0.392 ± 0.219	0.531 ± 0.151
yeast6	**0.974 ± 0.005**	0.227 ± 0.170	0.461 ± 0.093	0.364 ± 0.069
kr-vs-k-zero-vs-eight	**0.995 ± 0.003**	0.445 ± 0.216	0.876 ± 0.095	0.929 ± 0.081
abalone-20-vs-8-9-10	**0.982 ± 0.003**	0.255 ± 0.224	0.134 ± 0.054	0.254 ± 0.102
kddcup-buffer-overflow-vs-back	**1.000 ± 0.000**	0.983 ± 0.017	**1.000 ± 0.000**	**1.000 ± 0.000**
poker-8-vs-6	**0.984 ± 0.004**	0.244 ± 0.385	0.107 ± 0.003	0.140 ± 0.243
kddcup-rootkit-imap-vs-back	1.000 ± 0.001	0.981 ± 0.023	0.958 ± 0.033	**1.000 ± 0.000**

For the *Specificity* metric (Tab. 7.5), *SEMOOS* is the winner with a considerable advantage, reaching the metric close to 100%.

Despite presenting accurate metric values, choosing the best method is challenging. Therefore, it is necessary to perform statistical tests. The *Wilcoxon rank-sum test* was chosen for this purpose (Fig. 7.1). Each subsection (a-d) has five metrics and a comparison between the method listed at the top of the graph and the other methods on the left side of each small graph. The charts' green, yellow, and red colors mean to win, draw, and lose, respectively. The method wins with statistical significance if the green bar crosses the dashed line.

Table 7.3: *Precision*

DATASET	**MOOforest**	**SEMOOS**	**DE-Forest**	**MOLO-MDH**
glass1	**0.794 ± 0.033**	0.189 ± 0.179	0.788 ± 0.036	0.670 ± 0.048
haberman	**0.670 ± 0.043**	0.325 ± 0.326	0.643 ± 0.031	0.368 ± 0.057
yeast3	**0.935 ± 0.008**	0.806 ± 0.020	0.931 ± 0.009	0.750 ± 0.042
page-blocks0	**0.970 ± 0.003**	0.883 ± 0.036	0.970 ± 0.003	0.874 ± 0.018
yeast-2-vs-4	0.945 ± 0.010	0.868 ± 0.093	**0.947 ± 0.012**	0.720 ± 0.098
ecoli-0-6-7-vs-3-5	**0.946 ± 0.017**	0.778 ± 0.327	0.942 ± 0.013	0.699 ± 0.163
yeast-0-3-5-9-vs-7-8	**0.897 ± 0.013**	0.872 ± 0.137	0.884 ± 0.035	0.369 ± 0.051
yeast-0-2-5-6-vs-3-7-8-9	0.913 ± 0.007	0.830 ± 0.111	**0.919 ± 0.007**	0.621 ± 0.079
ecoli-0-1-vs-2-3-5	**0.953 ± 0.013**	0.883 ± 0.133	0.950 ± 0.012	0.708 ± 0.128
ecoli-0-2-6-7-vs-3-5	**0.935 ± 0.016**	0.826 ± 0.307	0.922 ± 0.038	0.684 ± 0.108
ecoli-0-6-7-vs-5	0.957 ± 0.017	**0.983 ± 0.050**	0.948 ± 0.018	0.744 ± 0.140
ecoli-0-1-4-7-vs-2-3-5-6	0.949 ± 0.009	0.912 ± 0.062	**0.951 ± 0.013**	0.631 ± 0.155
cleveland-0-vs-4	0.904 ± 0.034	0.781 ± 0.214	**0.919 ± 0.042**	0.521 ± 0.200
page-blocks-1-3-vs-4	**0.995 ± 0.004**	0.853 ± 0.175	0.991 ± 0.006	0.836 ± 0.118
abalone9-18	0.932 ± 0.013	0.822 ± 0.123	**0.933 ± 0.015**	0.479 ± 0.184
glass-0-1-6-vs-5	**0.961 ± 0.021**	0.582 ± 0.422	0.948 ± 0.031	0.561 ± 0.276
yeast-2-vs-8	0.974 ± 0.004	**1.000 ± 0.000**	0.959 ± 0.023	0.561 ± 0.107
yeast4	0.953 ± 0.008	0.610 ± 0.269	**0.954 ± 0.013**	0.379 ± 0.128
poker-9-vs-7	**0.963 ± 0.020**	0.000 ± 0.000	0.936 ± 0.000	0.256 ± 0.388
abalone-17-vs-7-8-9-10	0.966 ± 0.004	0.626 ± 0.260	**0.970 ± 0.005**	0.383 ± 0.082
abalone-21-vs-8	**0.976 ± 0.008**	0.842 ± 0.304	0.967 ± 0.014	0.603 ± 0.202
yeast6	0.972 ± 0.005	0.490 ± 0.388	**0.976 ± 0.005**	0.403 ± 0.095
kr-vs-k-zero-vs-eight	0.995 ± 0.004	0.825 ± 0.354	**0.995 ± 0.003**	0.919 ± 0.104
abalone-20-vs-8-9-10	**0.979 ± 0.004**	0.347 ± 0.293	0.974 ± 0.002	0.352 ± 0.141
kddcup-buffer-overflow-vs-back	**1.000 ± 0.000**	0.983 ± 0.050	**1.000 ± 0.000**	**1.000 ± 0.000**
poker-8-vs-6	**0.979 ± 0.006**	0.133 ± 0.299	0.977 ± 0.001	0.233 ± 0.396
kddcup-rootkit-imap-vs-back	1.000 ± 0.001	**1.000 ± 0.000**	0.999 ± 0.001	**1.000 ± 0.000**

The *Wilcoxon test* shows that *SEMOOS* (Fig. 7.1a) wins with statistical significance overall methods for the *Specificity* metric. In terms of other metrics, he has many losers. The *DE-Forest* method (Fig. 7.1b) wins regarding statistical significance in the *Recall* and *Precision* metrics over the *SEMOOS* and *MOLO-MDH* methods. It obtains good results for the *Gmean* metric. *MOOforest* (Fig. 7.1c has the most wins of all methods. It wins with statistical significance against at least one method for each metric. It achieves the best results for the *Gmean* metric. The last method *MOLO-MDH* (Fig. 7.1d) wins on many datasets in the *BAC* metric and, like *SEMOOS*, achieves good results in *Specificity*.

MOOforest is the best method for imbalanced data classification because it obtains the most promising results for the *Gmean* metric responsible for the minority class classification and the aggregated *BAC* metric. It is also a multi-objective method, so users can

Table 7.4: *Recall*

DATASET	**MOOforest**	**SEMOOS**	**DE-Forest**	**MOLO-MDH**
glass1	**0.795 ± 0.032**	0.055 ± 0.098	0.787 ± 0.031	0.650 ± 0.060
haberman	**0.706 ± 0.034**	0.047 ± 0.054	0.673 ± 0.025	0.351 ± 0.090
yeast3	**0.939 ± 0.007**	0.645 ± 0.065	0.936 ± 0.008	0.716 ± 0.055
page-blocks0	**0.971 ± 0.003**	0.550 ± 0.074	0.971 ± 0.003	0.842 ± 0.023
yeast-2-vs-4	0.947 ± 0.008	0.527 ± 0.108	**0.950 ± 0.010**	0.709 ± 0.118
ecoli-0-6-7-vs-3-5	**0.947 ± 0.018**	0.345 ± 0.185	0.941 ± 0.014	0.691 ± 0.136
yeast-0-3-5-9-vs-7-8	**0.909 ± 0.004**	0.136 ± 0.051	0.906 ± 0.010	0.316 ± 0.070
yeast-0-2-5-6-vs-3-7-8-9	0.923 ± 0.005	0.234 ± 0.078	**0.927 ± 0.005**	0.493 ± 0.056
ecoli-0-1-vs-2-3-5	**0.954 ± 0.012**	0.400 ± 0.153	0.948 ± 0.013	0.617 ± 0.125
ecoli-0-2-6-7-vs-3-5	**0.938 ± 0.013**	0.327 ± 0.164	0.930 ± 0.017	0.691 ± 0.093
ecoli-0-6-7-vs-5	**0.957 ± 0.015**	0.490 ± 0.187	0.950 ± 0.014	0.760 ± 0.136
ecoli-0-1-4-7-vs-2-3-5-6	0.951 ± 0.010	0.411 ± 0.183	**0.952 ± 0.009**	0.675 ± 0.147
cleveland-0-vs-4	0.926 ± 0.014	0.398 ± 0.227	**0.939 ± 0.015**	0.443 ± 0.172
page-blocks-1-3-vs-4	**0.995 ± 0.004**	0.393 ± 0.154	0.991 ± 0.007	0.871 ± 0.105
abalone9-18	**0.946 ± 0.007**	0.348 ± 0.111	0.946 ± 0.004	0.286 ± 0.085
glass-0-1-6-vs-5	0.958 ± 0.021	0.285 ± 0.244	**0.961 ± 0.018**	0.560 ± 0.304
yeast-2-vs-8	**0.974 ± 0.005**	0.450 ± 0.067	0.969 ± 0.009	0.470 ± 0.149
yeast4	0.963 ± 0.004	0.063 ± 0.031	**0.966 ± 0.003**	0.212 ± 0.070
poker-9-vs-7	**0.971 ± 0.010**	0.000 ± 0.000	0.967 ± 0.000	0.150 ± 0.200
abalone-17-vs-7-8-9-10	0.974 ± 0.003	0.176 ± 0.098	**0.976 ± 0.001**	0.252 ± 0.054
abalone-21-vs-8	0.977 ± 0.008	0.357 ± 0.194	**0.977 ± 0.006**	0.486 ± 0.183
yeast6	0.976 ± 0.004	0.081 ± 0.084	**0.979 ± 0.003**	0.332 ± 0.071
kr-vs-k-zero-vs-eight	0.995 ± 0.003	0.245 ± 0.169	**0.995 ± 0.003**	0.946 ± 0.109
abalone-20-vs-8-9-10	0.985 ± 0.002	0.115 ± 0.120	**0.986 ± 0.001**	0.185 ± 0.078
kddcup-buffer-overflow-vs-back	**1.000 ± 0.000**	0.967 ± 0.033	**1.000 ± 0.000**	**1.000 ± 0.000**
poker-8-vs-6	**0.989 ± 0.002**	0.221 ± 0.373	0.988 ± 0.001	0.086 ± 0.158
kddcup-rootkit-imap-vs-back	1.000 ± 0.001	0.964 ± 0.045	0.999 ± 0.001	**1.000 ± 0.000**

adapt it to their needs. Due to the experiments, we had to select one balanced solution from the solutions' set returned from *MOOforest* using the *PROMETHEE II* algorithm.

The *SEMOOS* method is also a multi-objective method. However, it uses the entire set of solutions the optimization algorithm returns to train the models, and then an ensemble is created from them. The solution was selected at the end by pruning models from the finished ensemble. This process contrasts the *MOLO* method, where the selection of possible solutions (created ensembles) was performed at each algorithm step using the *Beam search* function. This process allowed us to speed up the calculations and not consume memory with all possible combinations of individual models.

However, the purpose of the proposed methods was not to select a solution automatically but to show the various possibilities of the methods. The user can choose the more meaningful solution for him or her, which we did not choose in our experiments.

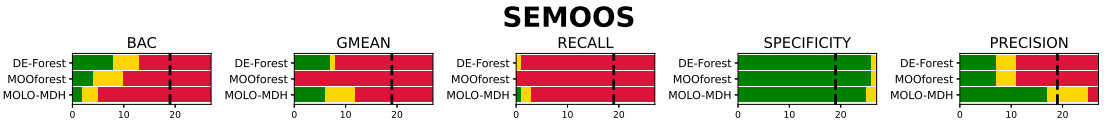
Table 7.5: *Specificity*

DATASET	**MOOforest**	**SEMOOS**	**DE-Forest**	**MOLO-MDH**
glass1	0.721 ± 0.046	**0.946 ± 0.055**	0.692 ± 0.039	0.822 ± 0.037
haberman	0.408 ± 0.050	**0.979 ± 0.025**	0.401 ± 0.041	0.786 ± 0.036
yeast3	0.673 ± 0.045	**0.981 ± 0.003**	0.583 ± 0.089	0.970 ± 0.007
page-blocks0	0.840 ± 0.030	**0.991 ± 0.004**	0.832 ± 0.026	0.986 ± 0.002
yeast-2-vs-4	0.628 ± 0.087	**0.990 ± 0.008**	0.652 ± 0.076	0.968 ± 0.012
ecoli-0-6-7-vs-3-5	0.679 ± 0.100	**0.988 ± 0.022**	0.557 ± 0.142	0.962 ± 0.026
yeast-0-3-5-9-vs-7-8	0.246 ± 0.043	**0.997 ± 0.003**	0.213 ± 0.060	0.940 ± 0.015
yeast-0-2-5-6-vs-3-7-8-9	0.439 ± 0.039	**0.993 ± 0.005**	0.433 ± 0.053	0.966 ± 0.012
ecoli-0-1-vs-2-3-5	0.639 ± 0.094	**0.993 ± 0.009**	0.549 ± 0.130	0.970 ± 0.015
ecoli-0-2-6-7-vs-3-5	0.588 ± 0.150	**0.994 ± 0.012**	0.442 ± 0.175	0.961 ± 0.021
ecoli-0-6-7-vs-5	0.699 ± 0.177	**0.999 ± 0.003**	0.617 ± 0.171	0.968 ± 0.023
ecoli-0-1-4-7-vs-2-3-5-6	0.609 ± 0.101	**0.995 ± 0.003**	0.551 ± 0.077	0.959 ± 0.021
cleveland-0-vs-4	0.305 ± 0.156	**0.989 ± 0.010**	0.301 ± 0.171	0.962 ± 0.019
page-blocks-1-3-vs-4	0.933 ± 0.052	**0.995 ± 0.006**	0.886 ± 0.074	0.988 ± 0.010
abalone9-18	0.259 ± 0.086	**0.994 ± 0.004**	0.174 ± 0.046	0.977 ± 0.014
glass-0-1-6-vs-5	0.615 ± 0.247	**0.992 ± 0.012**	0.391 ± 0.294	0.974 ± 0.025
yeast-2-vs-8	0.482 ± 0.156	**1.000 ± 0.000**	0.377 ± 0.240	0.983 ± 0.009
yeast4	0.173 ± 0.057	**0.999 ± 0.001**	0.125 ± 0.057	0.987 ± 0.006
poker-9-vs-7	0.274 ± 0.187	**1.000 ± 0.000**	0.033 ± 0.000	0.991 ± 0.018
abalone-17-vs-7-8-9-10	0.166 ± 0.060	**0.998 ± 0.002**	0.139 ± 0.053	0.989 ± 0.004
abalone-21-vs-8	0.442 ± 0.165	**0.999 ± 0.002**	0.205 ± 0.188	0.991 ± 0.006
yeast6	0.259 ± 0.086	**0.998 ± 0.002**	0.225 ± 0.094	0.987 ± 0.004
kr-vs-k-zero-vs-eight	0.778 ± 0.132	**1.000 ± 0.001**	0.779 ± 0.161	0.998 ± 0.002
abalone-20-vs-8-9-10	0.120 ± 0.084	**0.999 ± 0.001**	0.021 ± 0.023	0.995 ± 0.001
kddcup-buffer-overflow-vs-back	**1.000 ± 0.000**	1.000 ± 0.001	**1.000 ± 0.000**	**1.000 ± 0.000**
poker-8-vs-6	0.073 ± 0.185	0.985 ± 0.034	0.012 ± 0.001	**0.998 ± 0.002**
kddcup-rootkit-imap-vs-back	0.964 ± 0.060	**1.000 ± 0.000**	0.919 ± 0.063	**1.000 ± 0.000**

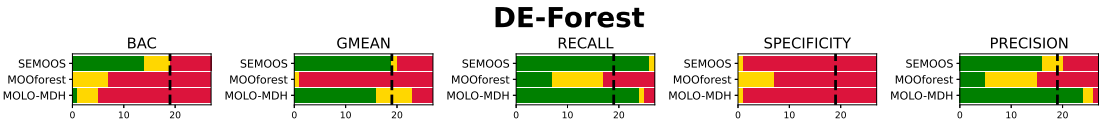
Depending on the prioritization of the criteria, the solution may be more tailored to the case each time.

Single-criteria methods usually aggregate metrics, which makes it impossible to obtain information about the exact values of individual components. The opposite is multi-objective methods that treat each criterion separately. This way, more easily accessible information, such as model preferences for specific classes, is retained. An example is the *Gmean* metric, i.e., the geometric mean of *Precision* and *Recall*. Single-criteria methods will use the aggregated *Gmean* metric, and multi-objective methods can treat *Precision* and *Recall* separately as two distinct criteria. The fitness functions can search for the minimum or maximum independently, which provides excellent opportunities for metrics other than classifier performance, e.g., cost metrics.

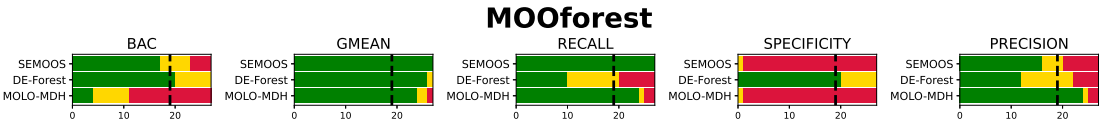
Choosing the most satisfactory method may be difficult or even impossible because



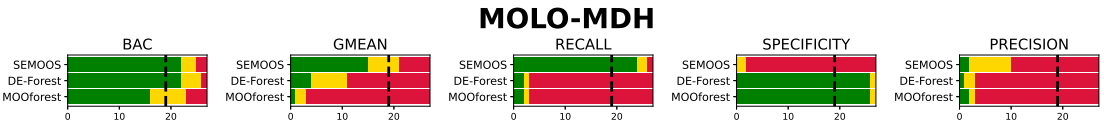
(a) SEMOOS vs other methods



(b) DE-Forest vs other methods



(c) MOOforest vs other methods



(d) MOLO-MDH vs other methods

Figure 7.1: *Wilcoxon ranking*

performance depends on many factors and, significantly, on the dataset. Referring to *Wolpert's "no free lunch"* theorem, there is no algorithm that would solve all problems with equally acceptable solution quality. Based on the results presented in this chapter, for instance, *MOLO-MDH* performs better in *Balanced Accuracy* with statistical significance than other methods for problems such as *yeast*, *ecoli*, or *abalone*. For most datasets, the *MOOforest* method has the best performance, but *DE-Forest* is more suitable at classifying highly imbalanced data, where there are few instances of the minority class, so it is essential to classify minority class well. Therefore, all the proposed methods in this work should be considered when designing AI systems because, depending on the specific decision-making problem, the methods should give promising results.

Chapter 8

Conclusion and future research directions

The thesis focused on the use of multi-objective optimization for employing feature selection to classify mainly imbalanced data. Feature selection determines a feature subspace for each model, and this mechanism ensures the diversity of models in the ensemble. Research in this area has shown that optimizing for feature selection gives good results, and the proposed methods sometimes outperform state-of-the-art methods. An additional advantage of the proposed algorithms using multi-objective optimization is the ability to select the most appropriate solution, which classical methods do not offer. The following research objectives confirmed the research hypothesis formulated at the beginning.

Using multi-objective optimization to train classifiers on feature subspaces produces models with no worse prediction quality than state-of-the-art methods and allows choosing a solution tailored to a user's needs.

Several objectives were formulated to prove the hypothesis. Let us discuss if they were achieved.

- **Development of feature selection methods based on Multi-Objective Optimization for constructing single classifiers.**

  This objective was met by proposing methods using optimization to perform feature selection. The methods use optimization algorithms such as *GA* in the single-criteria version and *NSGA-II* in the multi-objective version. The advantage of using optimization is taking into account not only the quality of the built classifier but also the cost of features, which is particularly important in cost-sensitive

learning. Simultaneous optimization of two criteria, maximizing performance and minimizing cost, in the case of multi-objective optimization, gives a set of solutions from which the user can choose the most suited to his needs. The proposed methods achieved comparable quality to classical methods, but the latter do not allow the possibility of choosing from a solution set and returning only one solution. Methods for feature selection in the classification task have been published in [56].

- **Development of a multi-objective method for training classifier ensembles using learning Support Vector Machines base classifiers on subspaces of the feature space.**

  This goal was achieved by proposing the *SEMOOS* method. *SEMOOS* is an ensemble consisting of single *SVM* classifiers using multi-objective optimization and the *NSGA-II* algorithm to search the feature space and find two parameters of *SVM* classifiers. *NSGA-II* returns a set of such solutions, and each of them is used to train a model, which is then added to the pool, creating an ensemble classifier. Using the proposed center-based bootstrapping and model pruning in the method is optional. The method was tested on many imbalanced datasets and achieved satisfactory results compared to reference methods. The *SEMOOS* method was published in [59].

- **Development of a method for training classifier ensemble using learning decision tree base classifiers on subspaces of the feature space and aggregated criteria.**

  The objective was completed by presenting the *DE-Forest* method using the *Differential Evolution* optimization algorithm to find the best feature vector for the entire ensemble relative to various aggregated metrics. Such a vector is appropriately prepared, and based on it, individual decision tree models can be trained to create an ensemble. The proposed method often outperforms state-of-the-art methods. The *DE-Forest* method was presented in [57].

- **Development of a multi-objective method for training classifier ensemble using learning decision tree base classifiers on subspaces of the feature space to form the non-random forest.**

  The goal was achieved by proposing the *MOOforest* method, an ensemble built from individual decision trees using the *MOEA/D* multi-objective optimization algorithm. *MOEA/D* searches the feature space for the entire ensemble based on two criteria simultaneously: *Precision* and *Recall*. Thanks to this, it returns a set of solutions from which the user can select one solution. Since we did not have access to users during the experiment and wanted to show that the obtained solutions do not differ qualitatively from state-of-the-art solutions that use single-objective

optimization, we used multi-criteria decision-making methods with predefined selection rules. The *MOOforest* method uses the *PROMETHEE II* function to select one solution and to be able to conduct experiments and compare with reference methods. The models are trained based on this solution that makes up the final ensemble. The proposed method, in many cases, outperforms the reference methods. The *MOOforest* method was published in [58].

- **Development of a multi-objective method for training classifier ensemble using learning base classifiers on subspaces of the feature space and local optimization.**

  The goal was achieved by proposing the *MOLO* method. It is a novel method using multi-objective local optimization to build a diverse ensemble. Each base DT model is trained on one bootstrapped subset. The optimization searches through possible solutions in each step and adds one model to the ensemble. The restrictions prevent the search from spreading to a considerable extent, and thanks to it, the algorithm selects the paths that provide the best results at a given moment. The *MOLO* method has been tested for two sets of dual criteria and can also handle the three criteria case. Finally, *MOLO* returns several ensembles from which the user can choose the best one for his needs. Extensive tests for imbalanced data comparing the proposed and reference methods showed *MOLO* advantage.

- **Experimental evaluation of the obtained methods.**

  This objective was achieved by comparing all proposed methods: *SEMOOS*, *DEForest*, *MOOforest*, and *MOLO*. Statistical tests and detailed results for each dataset showed each method's characteristics and competence areas.

Future works

The methods and research presented in the thesis can be extended in various ways:

- Future research on feature selection methods based on Multi-Objective Optimization may use other multi-objective optimization methods newer than *NSGA II*.

- Experiments for all proposed methods can be carried out for larger numbers of datasets. A precious experiment would be conducted for a specific real-world problem for which appropriate classification methods have yet to be found. Then, the proposed multi-objective approach could indicate solutions to the problem that would be more satisfactory for the user.

- It would be possible to test the methods for multi-class data. However, appropriate metrics must be used to measure the classification performance, and some metrics of the optimization criteria have to be changed.
- Future research could include examining many different optimization criteria within methods. Not only changing performance metrics but also various diversity metrics or metrics directly related to the data, such as feature cost.
- Most studies were conducted for only two criteria except the *MOLO* method. It would be worth conducting experiments for three or more criteria to have greater control over the final solution and adjustment to various criteria according to the user's priorities.
- The result of multi-objective methods is a non-dominated solution set, from which one solution is often selected. Trying other solution selection methods than *PROMETHEE II* or using more than one final solution would be worth testing.
- The proposed methods could use different optimization algorithms and classifiers.
- Comparison of the proposed methods could use more state-of-the-art methods.

Publications

Grzyb Joanna is my maiden name, which I changed at the end of 2023, so I now use the surname Klikowska Joanna.

- Grzyb, Joanna, Mariusz Topolski, and Michał Woźniak. "Application of multi-objective optimization to feature selection for a difficult data classification task." International Conference on Computational Science. Cham: Springer International Publishing, 2021.
  CORE: A, MNiSW: 140
- Grzyb, Joanna, Jakub Klikowski, and Michał Woźniak. "Hellinger distance weighted ensemble for imbalanced data stream classification." Journal of Computational Science 51 (2021): 101314.
  IF: 3.3, MNiSW: 100
- Klikowska, Joanna, Michał Woźniak. "Using multi-objective optimization to build non-random forest." Logic Journal of the IGPL, 2024. (after the first revision, where reviewers formulated minor comments only)
  IF: 1.0, MNiSW: 100

- Grzyb, Joanna, and Michał Woźniak. "SVM ensemble training for imbalanced data classification using multi-objective optimization techniques." Applied Intelligence 53.12 (2023): 15424-15441.
  IF: 5.3, MNiSW: 70

- Grzyb, Joanna, and Michał Woźniak. "DE-Forest–Optimized Decision Tree Ensemble." International Conference on Computational Collective Intelligence. Cham: Springer Nature Switzerland, 2023.
  CORE: C, MNiSW: 20

- Grzyb, Joanna, and Michał Woźniak. "MOOforest–Multi-objective Optimization to Form Decision Tree Ensemble." Proceedings of the XXI Polish Control Conference. Cham: Springer Nature Switzerland, 2023.
  MNiSW: 20

Acknowledgements

This work was supported by the Polish National Science Centre under the grant No. 2019/35/B/ST6/04442.

Bibliography

[1] H.A. Abbass. Pareto neuro-evolution: constructing ensemble of neural networks using multi-objective optimization. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, volume 3, pages 2074–2080 Vol.3, 2003.

[2] Abdiansah Abdiansah and Retantyo Wardoyo. Time complexity analysis of support vector machines (svm) in libsvm. *International Journal Computer and Application*, 128(3):28–34, 2015.

[3] Ajith Abraham and Lakhmi Jain. *Evolutionary Multiobjective Optimization*, pages 1–6. Springer London, London, 2005.

[4] Jesús Alcalá-Fdez, Alberto Fernández, Julián Luengo, Joaquín Derrac, Salvador García, Luciano Sánchez, and Francisco Herrera. KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17:255–287, 2011.

[5] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.

[6] Victor Henrique Alves Ribeiro and Gilberto Reynoso-Meza. Ensemble learning by means of a multi-objective optimization design approach for dealing with imbalanced data sets. *Expert Systems with Applications*, 147:113232, 2020.

[7] Farzana Anowar, Samira Sadaoui, and Bassant Selim. Conceptual and empirical comparison of dimensionality reduction algorithms (pca, kpca, lda, mds, svd, lle, isomap, le, ica, t-sne). *Computer Science Review*, 40:100378, 2021.

[8] Uroš Arsenijevic and Marija Jovic. Artificial intelligence marketing: Chatbots. In *2019 International Conference on Artificial Intelligence: Applications and Innovations (IC-AIAI)*, pages 19–193. IEEE, 2019.

[9] Shahrokh Asadi and Seyed Ehsan Roshan. A bi-objective optimization method to produce a near-optimal number of classifiers and increase diversity in bagging. *Knowledge-Based Systems*, 213:106656, 2021.

[10] Richard E Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[11] Urvesh Bhowan, Mark Johnston, Mengjie Zhang, and Xin Yao. Evolving diverse ensembles using genetic programming for classification with unbalanced data. *IEEE Transactions on Evolutionary Computation*, 17(3):368–386, 2013.

[12] Urvesh Bhowan, Mark Johnston, Mengjie Zhang, and Xin Yao. Reusing genetic programming for ensemble selection in classification of unbalanced data. *IEEE Transactions on Evolutionary Computation*, 18(6):893–908, 2014.

[13] Ying Bi, Bing Xue, and Mengjie Zhang. Multitask feature learning as multiobjective optimization: A new genetic programming approach to image classification. *IEEE Transactions on Cybernetics*, 53(5):3007–3020, 2023.

[14] J. Blank and K. Deb. Pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[15] Andrea Bommert, Xudong Sun, Bernd Bischl, Jörg Rahnenführer, and Michel Lang. Benchmark for filter methods for feature selection in high-dimensional classification data. *Computational Statistics and Data Analysis*, 143:106839, 2020.

[16] Paula Branco, Luís Torgo, and Rita P. Ribeiro. Relevance-based evaluation metrics for multi-class imbalanced domains. In Jinho Kim, Kyuseok Shim, Longbing Cao, Jae-Gil Lee, Xuemin Lin, and Yang-Sae Moon, editors, *Advances in Knowledge Discovery and Data Mining*, pages 698–710, Cham, 2017. Springer International Publishing.

[17] Jean-Pierre Brans and Yves De Smet. Promethee methods. In Salvatore Greco, Matthias Ehrgott, and José Rui Figueira, editors, *Multiple Criteria Decision Analysis: State of the Art Surveys*, pages 187–219, New York, NY, 2016. Springer New York.

[18] Jean-Pierre Brans and Bertrand Mareschal. The promcalc & gaia decision support system for multicriteria decision aid. *Decision Support Systems*, 12(4):297–310, 1994.

[19] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[20] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Routledge, 2017.

[21] Dariusz Brzezinski, Jerzy Stefanowski, Robert Susmaga, and Izabela Szczęch. Visual-based analysis of classification measures and their properties for class imbalanced problems. *Information Sciences*, 462:242 – 261, 2018.

[22] Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In Thanaruk Theeramunkong, Boonserm Kijsirikul, Nick Cercone, and Tu-Bao Ho, editors, *Advances in Knowledge Discovery and Data Mining*, pages 475–482, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[23] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, Jun 1998.

[24] Jie Cai, Jiawei Luo, Shulin Wang, and Sheng Yang. Feature selection in machine learning: A new perspective. *Neurocomputing*, 300:70–79, 2018.

[25] Leilei Cao, Lihong Xu, Erik D. Goodman, Chunteng Bao, and Shuwei Zhu. Evolutionary dynamic multiobjective optimization assisted by a support vector regression predictor. *IEEE Transactions on Evolutionary Computation*, 24(2):305–319, 2020.

[26] Arjun Chandra and Xin Yao. Ensemble learning using multi-objective evolutionary algorithms. *Journal of Mathematical Modelling and Algorithms*, 5(4):417–445, Dec 2006.

[27] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[28] Nitesh V. Chawla, Aleksandar Lazarevic, Lawrence O. Hall, and Kevin W. Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In Nada Lavrač, Dragan Gamberger, Ljupčo Todorovski, and Hendrik Blockeel, editors, *Knowledge Discovery in Databases: PKDD 2003*, pages 107–119, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[29] Xue-wen Chen and Michael Wasikowski. Fast: a roc-based feature selection metric for small samples and imbalanced data classification problems. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, page 124–132, New York, NY, USA, 2008. Association for Computing Machinery.

[30] K. R. Chowdhary. Natural language processing. In *Fundamentals of Artificial Intelligence*, pages 603–649, New Delhi, 2020. Springer India.

[31] David A Cieslak, T Ryan Hoens, Nitesh V Chawla, and W Philip Kegelmeyer. Hellinger distance decision trees are robust and skew-insensitive. *Data Mining and Knowledge Discovery*, 24(1):136–158, 2012.

[32] Carlos A. Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[33] Mwamba Kasongo Dahouda and Inwhee Joe. A deep-learned embedding technique for categorical features encoding. *IEEE Access*, 9:114381–114391, 2021.

[34] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31, 2011.

[35] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[36] Kalyanmoy Deb, Karthik Sindhya, and Tatsuya Okabe. Self-adaptive simulated binary crossover for real-parameter optimization. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, page 1187–1194, New York, NY, USA, 2007. Association for Computing Machinery.

[37] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning Research*, 7:1–30, dec 2006.

[38] Zhenyun Deng, Xiaoshu Zhu, Debo Cheng, Ming Zong, and Shichao Zhang. Efficient knn classification algorithm for big data. *Neurocomputing*, 195:143–148, 2016. Learning for Medical Imaging.

[39] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

[40] Tansel Dokeroglu, Ayça Deniz, and Hakan Ezgi Kiziloz. A comprehensive survey on recent metaheuristics for feature selection. *Neurocomputing*, 494:269–296, 2022.

[41] Laura Emmanuella A. dos S. Santana and Anne M. de Paula Canuto. Filter-based optimization techniques for selection of feature subsets in ensemble systems. *Expert Systems with Applications*, 41(4, Part 2):1622–1631, 2014.

[42] Matthias Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2005.

[43] Charles Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.

[44] Grandchamp Enguerran, Mohamed Abadi, and Olivier Alata. An hybrid method for feature selection based on multiobjective optimization and mutual information. *Journal of Informatics and Mathematical Sciences*, 7(1):21–48, 2015.

[45] Absalom E. Ezugwu, Abiodun M. Ikotun, Olaide O. Oyelade, Laith Abualigah, Jeffery O. Agushaka, Christopher I. Eke, and Andronicus A. Akinyelu. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, 110:104743, 2022.

[46] Nicolò Felicioni, Andrea Donati, Luca Conterio, Luca Bartoccioni, Davide Yi Xian Hu, Cesare Bernardis, and Maurizio Ferrari Dacrema. Multi-objective blended ensemble for highly imbalanced sequence aware tweet engagement prediction. In *Proceedings of the Recommender Systems Challenge 2020*, RecSysChallenge '20, page 29–33, New York, NY, USA, 2020. Association for Computing Machinery.

[47] Vitaliy Feoktistov. *Differential Evolution: In Search of Solutions*, volume 5. Springer Science & Business Media, 2007.

[48] Alberto Fernández, Cristobal José Carmona, María José del Jesus, and Francisco Herrera. A pareto-based ensemble with feature and instance selection for learning from multi-class imbalanced datasets. *International Journal of Neural Systems*, 27(06):1750028, 2017. PMID: 28633551.

[49] Alberto Fernández, María José del Jesus, and Francisco Herrera. Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced data-sets. *International Journal of Approximate Reasoning*, 50(3):561–577, 2009. Special Section on Bayesian Modelling.

[50] Alberto Fernández, Salvador García, María José del Jesus, and Francisco Herrera. A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets. *Fuzzy Sets and Systems*, 159(18):2378–2398, 2008.

[51] Sam Fletcher, Brijesh Verma, and Mengjie Zhang. A non-specialized ensemble classifier using multi-objective optimization. *Neurocomputing*, 409:93–102, 2020.

[52] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2012.

[53] Salvador Garcia and Francisco Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9(12), 2008.

[54] Salvador García and Francisco Herrera. Evolutionary Undersampling for Classification with Imbalanced Datasets: Proposals and Taxonomy. *Evolutionary Computation*, 17(3):275–306, 2009.

[55] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[56] Joanna Grzyb, Mariusz Topolski, and Michał Woźniak. Application of Multi-objective Optimization to Feature Selection for a Difficult Data Classification Task. In Maciej Paszynski, Dieter Kranzlmüller, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M.A. Sloot, editors, *Computational Science – ICCS 2021*, pages 81–94, Cham, 2021. Springer International Publishing.

[57] Joanna Grzyb and Michał Woźniak. DE-Forest – Optimized Decision Tree Ensemble. In Ngoc Thanh Nguyen, János Botzheim, László Gulyás, Manuel Núñez, Jan Treur, Gottfried Vossen, and Adrianna Kozierkiewicz, editors, *Computational Collective Intelligence*, pages 806–818, Cham, 2023. Springer Nature Switzerland.

[58] Joanna Grzyb and Michał Woźniak. MOOforest – Multi-objective Optimization to Form Decision Tree Ensemble. In Marek Pawelczyk, Dariusz Bismor, Szymon Ogonowski, and Janusz Kacprzyk, editors, *Advanced, Contemporary Control*, pages 108–117, Cham, 2023. Springer Nature Switzerland.

[59] Joanna Grzyb and Michał Woźniak. SVM ensemble training for imbalanced data classification using multi-objective optimization techniques. *Applied Intelligence*, 53(12):15424–15441, Jun 2023.

[60] Qiong Gu, Li Zhu, and Zhihua Cai. Evaluation measures of the classification performance of imbalanced data sets. In Zhihua Cai, Zhenhua Li, Zhuo Kang, and Yong Liu, editors, *Computational Intelligence and Intelligent Systems*, pages 461–471, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[61] Shenkai Gu, Ran Cheng, and Yaochu Jin. Multi-objective ensemble generation. *WIREs Data Mining and Knowledge Discovery*, 5(5):234–245, 2015.

[62] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In De-Shuang Huang, Xiao-Ping Zhang, and Guang-Bin Huang, editors, *Advances in Intelligent Computing*, pages 878–887, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[63] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[64] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[65] Pat Hudson. *The industrial revolution*. Bloomsbury Publishing, 2014.

[66] G. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1):55–63, 1968.

[67] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(03):90–95, may 2007.

[68] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, pages 507–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[69] H. Ishibuchi, Y. Nojima, and Tsutomu Doi. Comparison between single-objective and multi-objective genetic algorithms: Performance comparison and performance measures. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1143–1150, 2006.

[70] Hisao Ishibuchi and Yusuke Nojima. *Fuzzy Ensemble Design through Multi-Objective Fuzzy Rule Selection*, pages 507–530. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[71] Nathalie Japkowicz, Catherine Myers, and Mark Gluck. A novelty detection approach to classification. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'95, pages 518–523, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[72] Nathalie Japkowicz and Mohak Shah. *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.

[73] Yuchen Jiang, Xiang Li, Hao Luo, Shen Yin, and Okyay Kaynak. Quo vadis artificial intelligence? *Discover Artificial Intelligence*, 2(1):4, Mar 2022.

[74] Ruwang Jiao, Bach Hoai Nguyen, Bing Xue, and Mengjie Zhang. A survey on evolutionary multiobjective feature selection in classification: Approaches, applications, and challenges. *IEEE Transactions on Evolutionary Computation*, pages 1–1, 2023.

[75] Ruwang Jiao, Bing Xue, and Mengjie Zhang. Solving multi-objective feature selection problems in classification via problem reformulation and duplication handling. *IEEE Transactions on Evolutionary Computation*, pages 1–1, 2022.

[76] Dragi Kocev, Celine Vens, Jan Struyf, and Sašo Džeroski. Ensembles of multi-objective decision trees. In Joost N. Kok, Jacek Koronacki, Raomon Lopez de Mantaras, Stan Matwin, Dunja Mladenič, and Andrzej Skowron, editors, *Machine Learning: ECML 2007*, pages 624–631, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[77] Gang Kou, Pei Yang, Yi Peng, Feng Xiao, Yang Chen, and Fawaz E. Alsaadi. Evaluation of feature selection methods for text classification with small datasets using multiple criteria decision-making methods. *Applied Soft Computing*, 86:105836, 2020.

[78] Michał Koziarski, Bartosz Krawczyk, and Michał Woźniak. Radial-based approach to imbalanced data oversampling. In Francisco Javier Martínez de Pisón, Rubén Urraca, Héctor Quintián, and Emilio Corchado, editors, *Hybrid Artificial Intelligent Systems*, pages 318–327, Cham, 2017. Springer International Publishing.

[79] Michał Koziarski and Michał Woźniak. CCR: A combined cleaning and resampling algorithm for imbalanced data classification. *International Journal of Applied Mathematics and Computer Science*, 27(4), 2017.

[80] Bartosz Krawczyk, Michał Woźniak, and Bogusław Cyganek. Clustering-based ensembles for one-class classification. *Information Sciences*, 264:182–195, 2014. Serious Games.

[81] Bartosz Krawczyk, Michał Woźniak, and Gerald Schaefer. Cost-sensitive decision tree ensembles for effective imbalanced classification. *Applied Soft Computing*, 14:554–562, 2014.

[82] Ludmila I Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley Publishing, 2nd edition, 2014.

[83] Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, May 2003.

[84] Maciej Laszczyk and Paweł B. Myszkowski. Survey of quality measures for multi-objective optimization: Construction of complementary set of multi-objective quality measures. *Swarm and Evolutionary Computation*, 48:109–133, 2019.

[85] Hoang Lam Le, Dario Landa-Silva, Mikel Galar, Salvador Garcia, and I. Triguero. A hybrid surrogate model for evolutionary undersampling in imbalanced classification. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2020.

[86] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.

[87] Hui Li, Xi Yang, Yang Li, Li-Ying Hao, and Tian-Lun Zhang. Evolutionary extreme learning machine with sparse cost matrix for imbalanced learning. *ISA Transactions*, 100:198–209, 2020.

[88] Jinyan Li, Simon Fong, Raymond K. Wong, and Victor W. Chu. Adaptive multi-objective swarm fusion for imbalanced data classification. *Information Fusion*, 39:1–24, 2018.

[89] Jing Liang, Panpan Wei, Boyang Qu, Kunjie Yu, Caitong Yue, Yi Hu, and Shilei Ge. Ensemble learning based on multimodal multiobjective optimization. In Linqiang Pan, Jing Liang, and Boyang Qu, editors, *Bio-inspired Computing: Theories and Applications*, pages 299–313, Singapore, 2020. Springer Singapore.

[90] Jing Liang, Yuyang Zhang, Boyang Qu, Ke Chen, Kunjie Yu, and Caitong Yue. A multiform optimization framework for multi-objective feature selection in classification. *IEEE Transactions on Evolutionary Computation*, pages 1–1, 2023.

[91] Moshe Lichman et al. UCI Machine Learning Repository, 2013.

[92] Anping Lin, Peiwen Yu, Shi Cheng, and Lining Xing. One-to-one ensemble mechanism for decomposition-based multi-objective optimization. *Swarm and Evolutionary Computation*, 68:101007, 2022.

[93] Weiwei Lin, Ziming Wu, Longxin Lin, Angzhan Wen, and Jin Li. An ensemble random forest algorithm for insurance big data analysis. *IEEE Access*, 5:16568–16575, 2017.

[94] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022.

[95] Zhengyi Liu, Bo Chang, and Fan Cheng. An interactive filter-wrapper multi-objective evolutionary algorithm for feature selection. *Swarm and Evolutionary Computation*, 65:100925, 2021.

[96] Zhiming Lv, Linqing Wang, Zhongyang Han, Jun Zhao, and Wei Wang. Surrogate-assisted particle swarm optimization algorithm with pareto active learning for expensive multi-objective optimization. *IEEE/CAA Journal of Automatica Sinica*, 6(3):838–849, 2019.

[97] Victoria López, Alberto Fernández, Salvador García, Vasile Palade, and Francisco Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113–141, 2013.

[98] Victoria López, Alberto Fernández, Jose G. Moreno-Torres, and Francisco Herrera. Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. open problems on intrinsic data characteristics. *Expert Systems with Applications*, 39(7):6585–6608, 2012.

[99] Lianbo Ma, Min Huang, Shengxiang Yang, Rui Wang, and Xingwei Wang. An adaptive localized decision variable analysis approach to large-scale multiobjective and many-objective optimization. *IEEE Transactions on Cybernetics*, 52(7):6684–6696, 2022.

[100] Tomasz Maciejewski and Jerzy Stefanowski. Local neighbourhood extension of smote for mining imbalanced data. In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 104–111, 2011.

[101] Spyros Makridakis. The forthcoming Artificial Intelligence (AI) revolution: Its impact on society and firms. *Futures*, 90:46–60, 2017.

[102] Zbigniew Michalewicz. *Evolutionary Programming and Genetic Programming*, pages 283–287. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.

[103] Ingo Mierswa. Regularization through multi-objective optimization. In Alexander Hinneburg, editor, *LWA 2007: Lernen - Wissen - Adaption, Halle, Deutschland, September 2007, Workshop Proceedings*, pages 94–101. Martin-Luther-University Halle-Wittenberg, 2007.

[104] Amgad M. Mohammed, Enrique Onieva, Michał Woźniak, and Gonzalo Martínez-Muñoz. An analysis of heuristic metrics for classifier ensemble pruning based on ordered aggregation. *Pattern Recognition*, 124:108493, 2022.

[105] Kaustuv Nag and Nikhil R. Pal. Feature extraction and selection for parsimonious classifiers with multiobjective genetic programming. *IEEE Transactions on Evolutionary Computation*, 24(3):454–466, 2020.

[106] Krystyna Napierala and Jerzy Stefanowski. Identification of different types of minority class examples in imbalanced data. In Emilio Corchado, Václav Snášel, Ajith Abraham, Michał Woźniak, Manuel Graña, and Sung-Bae Cho, editors, *Hybrid Artificial Intelligent Systems*, pages 139–150, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[107] Krystyna Napierala and Jerzy Stefanowski. Types of minority class examples and their influence on learning classifiers from imbalanced data. *Journal of Intelligent Information Systems*, 46:563–597, 2016.

[108] Bach Hoai Nguyen, Bing Xue, Peter Andreae, Hisao Ishibuchi, and Mengjie Zhang. Multiple reference points-based decomposition for multiobjective feature selection in classification: Static and dynamic mechanisms. *IEEE Transactions on Evolutionary Computation*, 24(1):170–184, 2020.

[109] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[110] Luiz S. Oliveira, Marisa Morita, Robert Sabourin, and Flávio Bortolozzi. Multi-objective genetic algorithms to create ensemble of classifiers. In Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization*, pages 592–606, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[111] Aytuğ Onan, Serdar Korukoğlu, and Hasan Bulut. A multiobjective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification. *Expert Systems with Applications*, 62:1–16, 2016.

[112] Serafim Opricovic. Multicriteria optimization of civil engineering systems. *Faculty of Civil Engineering, Belgrade*, 2(1):5–21, 1998.

[113] Jason Papathanasiou and Nikolaos Ploskas. Multiple criteria decision aid. *Methods, Examples and Python Implementations*, 136:131, 2018.

[114] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[115] Wenbin Pei, Bing Xue, Mengjie Zhang, Lin Shang, Xin Yao, and Qiang Zhang. A survey on unbalanced classification: How can evolutionary computation help? *IEEE Transactions on Evolutionary Computation*, pages 1–1, 2023.

[116] Noraini Mohd Razali and John Geraghty. Genetic algorithm performance with different selection strategies in solving TSP. In *Proceedings of the World Congress*

on Engineering, volume 2, pages 1–6. International Association of Engineers Hong Kong, 2011.

[117] Beatriz Remeseiro and Veronica Bolon-Canedo. A review of feature selection methods in medical applications. *Computers in Biology and Medicine*, 112:103375, 2019.

[118] Victor Henrique Alves Ribeiro and Gilberto Reynoso-Meza. A multi-objective optimization design framework for ensemble generation. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '18, page 1882–1885, New York, NY, USA, 2018.

[119] Diah Risqiwati, Adhi Dharma Wibawa, Evi Septiana Pane, Wardah Rahmatul Islamiyah, Agnes Estuning Tyas, and Mauridhi Hery Purnomo. Feature Selection for EEG-Based Fatigue Analysis Using Pearson Correlation. In *2020 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, pages 164–169, 2020.

[120] Miao Rong, Dunwei Gong, Witold Pedrycz, and Ling Wang. A multimodel prediction method for dynamic multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 24(2):290–304, 2020.

[121] Bart Ian Rylander. *Computational complexity and the genetic algorithm*. University of Idaho, 2001.

[122] Thomas L. Saaty. Decision making with the analytic hierarchy process. *International Journal of Services Sciences*, 1(1):83–98, 2008.

[123] Kshira Sagar Sahoo, Bata Krishna Tripathy, Kshirasagar Naik, Somula Ramasubbareddy, Balamurugan Balusamy, Manju Khari, and Daniel Burgos. An Evolutionary SVM Model for DDOS Attack Detection in Software Defined Networks. *IEEE Access*, 8:132502–132513, 2020.

[124] Habiba Muhammad Sani, Ci Lei, and Daniel Neagu. Computational complexity analysis of decision tree algorithms. In Max Bramer and Miltos Petridis, editors, *Artificial Intelligence XXXV*, pages 191–197, Cham, 2018. Springer International Publishing.

[125] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[126] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Commun. ACM*, 63(12):54–63, nov 2020.

[127] Mohammad Shehab, Laith Abualigah, Qusai Shambour, Muhannad A. Abu-Hashem, Mohd Khaled Yousef Shambour, Ahmed Izzat Alsalibi, and Amir H. Gandomi. Machine learning in medical applications: A review of state-of-the-art methods. *Computers in Biology and Medicine*, 145:105458, 2022.

[128] Utkarsh Singh and Shyam Narain Singh. Optimal Feature Selection via NSGA-II for Power Quality Disturbances Classification. *IEEE Transactions on Industrial Informatics*, 14(7):2994–3002, 2018.

[129] Christopher Smith and Yaochu Jin. Evolutionary multi-objective generation of recurrent neural network ensembles for time series prediction. *Neurocomputing*, 143:302–311, 2014.

[130] Paolo Soda. A multi-objective optimisation approach for class imbalance learning. *Pattern Recognition*, 44(8):1801 – 1810, 2011.

[131] Bernd Carsten Stahl and Damian Eke. The ethics of ChatGPT – Exploring the ethical issues of an emerging technology. *International Journal of Information Management*, 74:102700, 2024.

[132] Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Springer Science & Business Media, 2008.

[133] Katarzyna Stąpor, Paweł Ksieniewicz, Salvador García, and Michał Woźniak. How to design the fair experimental classifier evaluation. *Applied Soft Computing*, 104:107219, 2021.

[134] Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, Dec 1997.

[135] Ikram Sumaiya Thaseen and Cherukuri Aswani Kumar. Intrusion detection model using fusion of chi-square feature selection and multi class SVM. *Journal of King Saud University - Computer and Information Sciences*, 29(4):462–472, 2017.

[136] Jiliang Tang, Salem Alelyani, and Huan Liu. Feature selection for classification: A review. *Data Classification: Algorithms and Applications*, page 37, 2014.

[137] Akbar Telikani, Amirhessam Tahmassebi, Wolfgang Banzhaf, and Amir H. Gandomi. Evolutionary machine learning: A survey. *ACM Comput. Surv.*, 54(8), oct 2021.

[138] Alaa Tharwat. Parameter investigation of support vector machine classifier with kernel functions. *Knowledge and Information Systems*, 61(3):1269–1302, Dec 2019.

[139] Ye Tian, Langchun Si, Xingyi Zhang, Ran Cheng, Cheng He, Kay Chen Tan, and Yaochu Jin. Evolutionary large-scale multi-objective optimization: A survey. *ACM Comput. Surv.*, 54(8), oct 2021.

[140] Michał K. Tomczyk and Miłosz Kadziński. Decomposition-based interactive evolutionary algorithm for multiple objective optimization. *IEEE Transactions on Evolutionary Computation*, 24(2):320–334, 2020.

[141] Mariusz Topolski and Marcin Beza. Modification of the principal component analysis method based on feature rotation by class centroids. *JUCS: Journal of Universal Computer Science*, 28(3), 2022.

[142] Isaac Triguero, Sergio González, Jose M. Moyano, Salvador García, Jesús Alcalá-Fdez, Julián Luengo, Alberto Fernández, Maria José del Jesús, Luciano Sánchez, and Francisco Herrera. KEEL 3.0: An Open Source Software for Multi-Stage Analysis in Data Mining. *International Journal of Computational Intelligence Systems*, 10(1):1238–1249, Jan 2017.

[143] Muhammad Usman and Huanhuan Chen. Pro-IDD: Pareto-based ensemble for imbalanced and drifting data streams. *Knowledge-Based Systems*, 282:111103, 2023.

[144] Vladimir Vapnik. *Estimation of dependences based on empirical data.* Springer Science & Business Media, 2006.

[145] B. Venkatesh and J. Anuradha. A review of feature selection and its methods. *Cybernetics and Information Technologies*, 19(1):3–26, 2019.

[146] Peng Wang, Bing Xue, Jing Liang, and Mengjie Zhang. Differential evolution-based feature selection: A niching-based multiobjective approach. *IEEE Transactions on Evolutionary Computation*, 27(2):296–310, 2023.

[147] S. Wang, H. Chen, and X. Yao. Negative correlation learning for classification ensembles. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2010.

[148] Xianpeng Wang, Yao Wang, Lixin Tang, and Qingfu Zhang. Multi-objective ensemble learning with multi-scale data for product quality prediction in iron and steel industry. *IEEE Transactions on Evolutionary Computation*, pages 1–1, 2023.

[149] Wes 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*, volume 445, pages 56 – 61, 2010.

[150] Weronika Węgier, Michał Koziarski, and Michał Woźniak. Multicriteria classifier ensemble learning for imbalanced data. *IEEE Access*, 10:16807–16818, 2022.

[151] Szymon Wojciechowski. Multi-objective evolutionary undersampling algorithm for imbalanced data classification. In Maciej Paszynski, Dieter Kranzlmüller, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M.A. Sloot, editors, *Computational Science – ICCS 2021*, pages 118–127, Cham, 2021. Springer International Publishing.

[152] Michał Woźniak, Manuel Graña, and Emilio Corchado. A survey of multiple classifier systems as hybrid systems. *Inf. Fusion*, 16:3–17, March 2014.

[153] Michał Woźniak. *Hybrid classifiers: methods of data, knowledge, and classifier combination*, volume 519. Springer, 2013.

[154] Yu Wu, Yongshan Zhang, Xiaobo Liu, Zhihua Cai, and Yaoming Cai. A multiobjective optimization-based sparse extreme learning machine algorithm. *Neurocomputing*, 317:88–100, 2018.

[155] Hang Xu, Bing Xue, and Mengjie Zhang. Segmented initialization and offspring modification in evolutionary algorithms for bi-objective feature selection. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, page 444–452, New York, NY, USA, 2020.

[156] Xiaozhan Xu. The sir method: A superiority and inferiority ranking method for multiple criteria decision making. *European Journal of Operational Research*, 131(3):587–602, 2001.

[157] Jiali Yan, Kristin A. Linn, Brian W. Powers, Jingsan Zhu, Sachin H. Jain, Jennifer L. Kowalski, and Amol S. Navathe. Applying machine learning algorithms to segment high-cost patient populations. *Journal of General Internal Medicine*, 34(2):211–217, Feb 2019.

[158] Shangshang Yang, Ye Tian, Cheng He, Xingyi Zhang, Kay Chen Tan, and Yaochu Jin. A gradient-guided evolutionary approach to training deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2021.

[159] Xin-She Yang. *Nature-inspired optimization algorithms*. Academic Press, 2020.

[160] Yaochu Jin, T. Okabe, and B. Sendhoff. Neural network regularization and ensembling using multi-objective evolutionary algorithms. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 1, pages 1–8 Vol.1, June 2004.

[161] K Yoon. Multiple attributes decision making methods and applications. *Springer-Verlag, Nowy Jork*, 1981.

[162] Yusliza Yusoff, Mohd Salihin Ngadiman, and Azlan Mohd Zain. Overview of NSGA-II for Optimizing Machining Process Parameters. *Procedia Engineering*, 15:3978–3983, 2011. CEIS 2011.

[163] Rizgar Zebari, Adnan Abdulazeez, Diyar Zeebaree, Dilovan Zebari, and Jwan Saeed. A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. *Journal of Applied Science and Technology Trends*, 1(2):56–70, 2020.

[164] Nana Zhang, Shi Ying, Weiping Ding, Kun Zhu, and Dandan Zhu. WGNCS: A robust hybrid cross-version defect model via multi-objective optimization and deep enhanced feature representation. *Information Sciences*, 570:545–576, 2021.

[165] Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[166] Qingyang Zhang, Shengxiang Yang, Shouyong Jiang, Ronggui Wang, and Xiaoli Li. Novel prediction strategies for dynamic multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 24(2):260–274, 2020.

[167] Xin Zhang and Wang Dahu. Application of artificial intelligence algorithms in image processing. *Journal of Visual Communication and Image Representation*, 61:42–49, 2019.

[168] Yu Zhou, Yan Qiu, and Sam Kwong. Region purity-based local feature selection: A multiobjective perspective. *IEEE Transactions on Evolutionary Computation*, 27(4):787–801, 2023.

[169] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[170] Zhi-Hua Zhou. *Machine learning*. Springer Nature, 2021.

[171] Zhi-Hua Zhou and Xu-Ying Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):63–77, 2006.