WROCLAW UNIVERSITY OF SCIENCE AND TECHNOLOGY

Application of multi-objective optimisation methods in the imbalanced data problem

by

Weronika Węgier

Supervisor:
prof. dr hab. inż. Michał Woźniak
Assistant supervisor:
dr inż. Michał Koziarski

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

November 2025

Acknowledgements

I am sincerely grateful to my two cats, Jaskier and Alistair, for making sure I get up every day to work on this dissertation. Some may say it was because they were hungry, but I truly believe that they wanted a person with appropriate qualifications to clean their litter box.

Abbreviations

ADASYN	**ADA**ptive **SYN**thetic Sampling
AUC	**A**rea **U**nder the Receiver Operating **C**urve
BAC	**B**alanced **AC**curacy score
CART	**C**lassification **A**nd **R**egression **T**ree
CCR	**C**ombined **C**learning and **R**esampling
COSMOS	**C**onditioned **O**ne-**S**hot **M**ulti-**O**bjective **S**earch
GA	**G**enetic Algorithm
IR	**I**mbalance **R**atio
MCDM	**M**ulti-**C**riteria **D**ecision **M**aking
MGDA	**M**ultiple-**G**radient **D**escent **A**lgorithm
MLP	**M**ulti-**L**ayer **P**erceptron
MOO	**M**ulti-**O**bjective **O**ptimisation
NSGA II	**N**on-dominated **S**orting **G**enetic **A**lgorithm II
ParetoMTL	**Pareto** **M**ulti-**T**asking **L**earning
PROMETHEE	**P**reference **R**anking **O**rganisation **M**ethod for **E**nrichment **E**valuations
ROS	**R**andom **O**ver**S**ampler
SMOTE	**S**ynthetic **M**inority **O**versampling **TE**chnique
SOO	**S**ingle-**O**bjective **O**ptimisation
XAI	E**X**plainable **A**rtificial **I**ntelligence

Symbols

b	number of bags
c	number of problem's classes
c_i	centroid of the cluster
C_i	the i-th cluster
$d(*)$	a distance measure
D	mapping representing classifier
f_i	fractions of all samples generated in the i-th sphere
g	discriminant function
$K(*)$	kernel function
$l(*)$	loss function
$\mathcal{LS}$	learning set
$P(*)$	probability of the event
Π	pool of estimators in ensemble
ϕ	activation function
Ψ	classification algorithm
Ψ_i	the i-th member of an ensemble of classifiers
r_i	radius of the i-th sphere
$\mathbb{R}^n$	feature space
s	number of samples in a subset

$\mathcal{TM}$	set of learning methods
$\mathcal{TM}_i$	learning algorithm
$\mathcal{TS}$	testing set
u_i	logical value determining whether the i-th sample is removed from the dataset
$\varphi(*)$	mapping function used by support vector machines
vip_i	importance of the i-th feature
x	feature vector representing the object in pattern recognition problem and a solution in an opt
X	feature space
$\mathcal{X}$	solution space in optimisation problem
X_{maj}	set of majority class samples
X_{min}	set of minority class samples
y	true label
$\hat{y}$	model's response / predicted label
$[*]$	Inverson's bracket

Abstract

The imbalanced data pose a significant challenge in the pattern recognition task. When not countered, it may lead to poor prediction quality and strong bias towards the more numerous class, which is also usually of less importance. One of the reasons for this difficulty is the way of quality assessment. Traditional measures or objective (loss) functions usually assume an equal cost of misclassification for each class. This, together with object number disproportion, may result in a bigger focus on majority class prediction or even omitting the minority class entirely. The usual solution to this problem - using metrics aggregating classes' recognition quality in a more balanced manner- faces the concern of unclear optimisation direction, as the score itself does not hold information about its factors, which reflect the performance of the problem's classes prediction. The answer to that problem could be multi-objective optimisation (MOO), which allows maximising classification quality for each class simultaneously.

This dissertation studies the application of multi-objective optimisation techniques in methods dedicated to imbalanced data classification. The aim of the research is to substantiate the hypothesis that ***incorporating MOO in training imbalanced data classifiers allows obtaining tailored solutions whose quality is no worse than using single-objective optimisation***. The following research questions were formulated and answered to support this claim:

1. **Is it feasible to employ MOO in the process of training of the ensemble classifier, and how does it compare to the ensembles optimised using a single criterion?**

   The ensemble method was proposed, employing the MOO algorithm in assigning weights to committee members. The experiments showed the advantage of utilising multiple criteria over models, where weights were optimised based on single objectives - both simple and aggregated quality metrics.

2. **Is it possible to employ MOO in the preprocessing stage, and how does it improve the quality of a classification model**

The question was answered by a proposition of a hybrid sampling algorithm, where parameters of sampling areas were optimised utilising the NSGA II method. The conducted research compared the proposed approach with a classifier trained on original data with no preprocessing, as well as popular sampling techniques.

3. **What is the best approach to estimate the quality criteria of the classifiers built using MOO?**

   The study was conducted by comparing three different estimation protocols - hold-out, testing on the training set, and 5x2 cross-validation. The results were analysed in terms of the quality of generated Pareto front estimations, the consistency of performance on training and validation data, and the overall quality of the obtained solutions.

4. **Is it possible to employ MOO gradient methods for the imbalanced data problem?**

   To answer the question, the utilisation of the weighted cross-entropy was proposed, aiming to substitute for previously employed quality metrics. The series of experiments was conducted to determine their applicability to the imbalanced data problem, as well as their relation with the target criteria.

5. **What is the diversity of classifiers from the Pareto front?**

   The aspect of diversity was considered as a means to support a solution selection. The study focused on the interpretability of the models originated from MOO results, and two approaches were proposed as a way to analyse the Pareto fronts.

Streszczenie

Problem danych niezbalansowanych stanowi znaczące wyzwanie w zadaniu rozpoznawania wzorców. Bez podjęcia odpowiednich środków może prowadzić do słabych zdolności predykcyjnych oraz preferowania decyzji związanych z klasą większościową. Jedną z trudności tego problemu jest wybór odpowiedniego sposobu oceny jakości modeli. Tradycyjne miary oraz funkcje celu (bądź straty) zakładają równy koszt nieprawidłowej klasyfikacji próbek, co wraz z dysproporcją między obiektami może skutkować większym naciskiem na poprawną predykcję klasy większościowej, a nawet całkowitym pominięciem klasy mniejszościowej. Standardowym rozwiązaniem jest wykorzystanie metryk agregujących jakość rozpoznawania poszczególnych klas. W wyniku agregacji tracone są jednak wartości składowych odpowiedzialnych za predykcję konkretnych klas, co skutkuje brakiem kontroli nad kierunkiem optymalizacji modelu. Odpowiedzią na to mogłoby być zastosowanie optymalizacji wielokryterialnej, która pozwala na jednoczesną poprawę rozpoznawania wszystkich klas.

Poniższa rozprawa skupia się na analizie wykorzystania optymalizacji wielokryterialnej w metodach przeznaczonych do rozwiązania problemu danych niezbalansowanych. Celem rozprawy jest uprawdopodobnienie hipotezy, że ***włączenie optymalizacji wielokryterialnej w trenowanie klasyfikatorów danych niezbalansowanych pozwala na osiągnięcie rozwiązań dopasowanych do potrzeb użytkownika, a także o jakości nie gorszej niż metody wykorzystujące optymalizacje jednokryterialną***. Następujące pytania badawcze zostały sformułowane oraz zaadresowane w celu podparcia powyższego twierdzenia:

1. **Czy jest możliwe wykorzystanie optymalizacji wielokryterialnej w uczeniu zespołu klasyfikatorów oraz jak wypada w porównaniu do zespołów optymalizowanych przy użyciu pojedynczej funkcji celu?**

   W pracy zaproponowano metodę wykorzystującą optymalizację wielokryterialną do przydzielania wag członkom zespołu klasyfikatorów. Eksperymenty wykazały przewagę w zastosowaniu wielu kryteriów nad modelami, których wagi optymalizowane były zgodnie z prostymi i zagregowanymi miarami oceny jakości.

2. **Czy jest możliwym zastosowanie optymalizacji wielokryterialnej na etapie przetwarzania wstępnego danych oraz jak to poprawia jakość klasyfikacji?**

   Odpowiedzią na pytanie była propozycja hybrydowego algorytmu samplującego, którego parametry obszarów próbkowania były optymalizowane przy pomocy metody NSGA II. Przeprowadzone badania porównały zaproponowane podejście z klasyfikatorem wyuczonym na oryginalnych danych, a także poddanych przetwarzaniu wstępnemu z wykorzystaniem popularnych technik próbkowania.

3. **Jaki jest najlepszy sposób na estymację kryteriów jakości klasyfikatorów trenowanych przy pomocy optymalizacji wielokryterialnej?**

   Przeprowadzono badanie porównujące trzy różne protokoły estymacji jakości modelu - hold-out, testowanie na zbiorze trenującym oraz walidację krzyżową 5x2. Wyniki zostały przeanalizowane pod kątem jakości estymowanych frontów Pareto, zgodności wyników otrzymanych na danych testowych i walidacyjnych, a także ogólnej jakości otrzymanych rozwiązań.

4. **Czy jest możliwe zastosowanie metod optymalizacji wielokryterialnej opartych na gradientach w problemie danych niezbalansowanych?**

   Żeby odpowiedzieć na pytanie, zostało zaproponowane użycie ważonej entropii krzyżowej, służącej do zastąpienia wcześniej wykorzystywanych metryk oceny jakości. Seria eksperymentów została przeprowadzona w celu określenia jej użyteczności w problemie danych niezbalansowanych, a także zgodności z kryteriami docelowymi.

5. **Jaki jest poziom różnorodności klasyfikatorów z frontu Pareto?**

   Rozważono różnorodność frontu Pareto w celu wsparcia procesu wyboru rozwiązania. Wywód skupił się na interpretowalności modeli powstałych w wyniku optymalizacji wielokryterialnej i zostały zaproponowane dwa podejścia do analizy frontu Pareto.

Contents

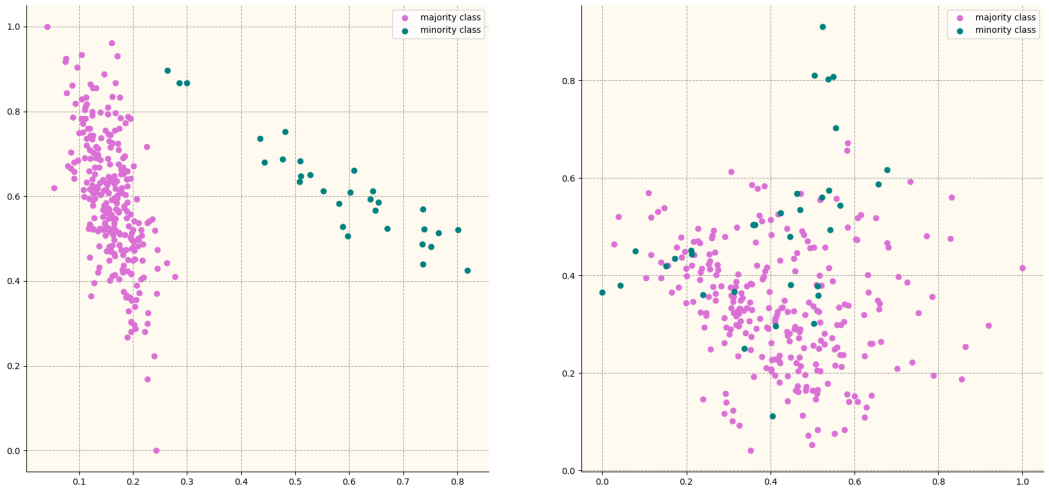
Chapter 1

Introduction

Since the beginning of time, people have used data obtained from different observations and measurements to learn about the surrounding world. The gathered information about plants' characteristics, celestial bodies' movements or specific symptoms allowed humans to prevent poisoning, predict eclipses or diagnose diseases. Such utilisation of the data required prior knowledge about the phenomenon bases, for example, in the form of sets of rules (i.e. *bitter taste may indicate that the plant is poisonous*) or more complex mathematical models. These, in turn, demanded long-standing experiments or measurements together with assessing numerous hypotheses. Still, some dependencies were too complex or difficult for humans to deduce.

The construction of computers and the development of computer science gave rise to machine learning - the discipline meant to notice the patterns in data for various reasons that are hard to find or even invisible to the human eye. Various proposed models aim to differentiate the objects according to some principles, for example, their types, or *classes* - i.e., *poisonous* or *safe* plant. They use different approaches and assumptions to optimise their parameters according to some objective function and thus improve the prediction quality. However, according to Wolpert's *no free lunch* theorem [1], no machine learning model is predominant for every problem. Therefore, depending on the nature of the data, the proper method needs to be chosen to obtain the best results. Moreover, the problems are often marked by additional difficulties which require even more careful model selection and implementation of specific prevention mechanisms.

One of such challenges is presented in the *imbalanced data problem*. The problem occurs when there is a disproportion between the sizes of the classes. In a binary classification task, the class with more samples is called *majority class*, and the one with a smaller population is called *minority class*. The reason behind the difficulty of classifying imbalanced data stems from the way how most of the models' objective function is formulated.



(a) *An easy problem* **(b)** *A hard problem*

Figure 1.1: *Examples of problems with the disproportion of the class sizes.*

Usually, the classical machine learning methods assume that the error on each sample is equivalently significant, which may lead to a strong bias towards the majority class, as reducing its prediction error decreases the overall error to a higher degree. Moreover, it is not always the number of class samples that poses a challenge, but their distribution in the feature space [2] (Figure 1.1). In a worst-case scenario, it may result in the model always predicting a sample as belonging to the majority class. However, usually, it is a prediction of minority class objects that is crucial, especially in domains such as medicine [3], bank security [4] or computer networks [5]. Very often, in the case of rare illness diagnosis, card fraud prevention or cyber attack detection, the rarer instances (i.e., sick patients) are far more important to correctly classify, as their omitting may result in high human and financial costs. The solution to that could be to always treat the sample as belonging to the more significant class, but it would also be too costly and may result in unnecessary treatments or preemptory measures.

One of the problems with tackling imbalanced data is the selection of a proper optimisation criterion. As mentioned earlier, the classical prediction error, or complementary prediction accuracy, does not suit data with class disproportion, as it often leads to a bias towards the majority class. There are base quality metrics used for imbalanced data, such as *precision* or *recall*. However, they show only one of the problem classes' prediction quality, mostly disregarding the others, and thus tend to lead to the bias towards either of the problem's classes. On the other hand, aggregated measures, for example, *balanced accuracy* or *geometric mean*, though taking all of the aspects under consideration, lose information about each of the model's properties since they are different functions of the base metrics [6] (Figure 1.2), and do not allow to steer the model's quality into the o desired direction.

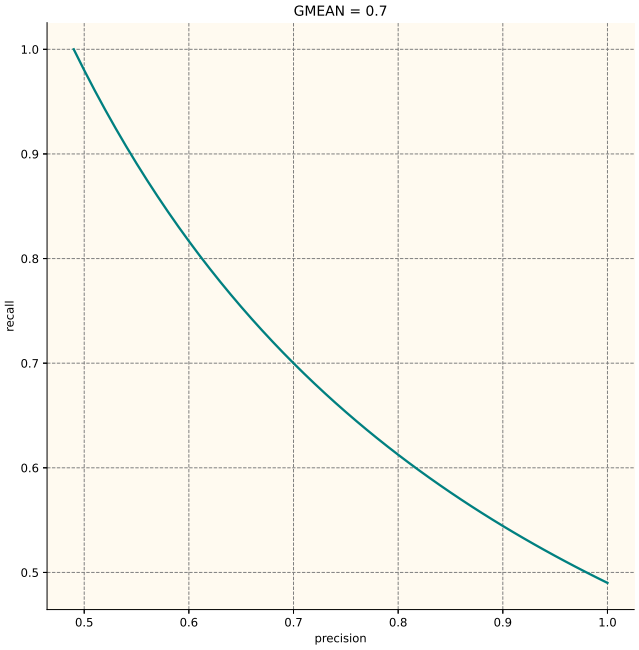


Figure 1.2: *Different values of precision and recall resulting in the same* geometric mean *score.*

This thesis proposition of solution to that problem is the employment of multi-objective optimisation (MOO) algorithms to the training of the imbalanced data classifiers. Optimisation of multiple criteria allows to overcome the problem of focusing on one aspect of the model's quality, as well as the information loss and randomness of the aggregated measures. By optimising base metrics corresponding to each of the classes' recognition, we can ensure that the model is not biased towards either of them. Furthermore, since MOO optimisation algorithms result in several solutions, there is a possibility to choose the predictive model configuration that fits the user's needs the best.

In compliance with the above motivation, the main research hypothesis is to prove that

Incorporating MOO in training imbalanced data classifiers allows obtaining customised solutions which quality is no worse than using single-objective optimisation.

To make the research hypothesis probable, the following research question will be answered in this dissertation:

1. Is it feasible to employ MOO in the process of training of the ensemble classifier, and how does it compare to the ensembles optimised using a single criterion?
2. Is it possible to employ MOO in the preprocessing stage, and how does it improve the quality of a classification model?

3. What is the best approach to estimate the quality criteria of the classifiers built using MOO?
4. Is it possible to employ MOO gradient methods for the imbalanced data problem?
5. What is the diversity of classifiers from the Pareto front?

The structure of this dissertation is as follows. Chapter 2 describes the background and related works of the task of machine learning, the imbalanced data problem, multi-objective optimisation and its applications in the aforementioned. Chapter 3 presents the first proposed method - the ensemble of classifiers employing MOO to weights optimisation. Chapter 4 includes the second proposed approach, where MOO is utilised in a parameter tuning of a sampling algorithm. Chapter 5 shows the analysis of the application of different evaluation protocols in MOO objectives estimation. Chapter 6 proposes a utilisation of weighted cross-entropies as surrogate criteria for gradient MOO algorithms and assesses its correspondence to previously used objectives. Chapter 7 examines the selection of solutions from the estimation of a Pareto front in the context of analysing and understanding the resulting classification models. Lastly, Chapter 8 concludes the conducted research and outlines possible future directions.

Chapter 2

Related works

This chapter describes the state of knowledge in the areas related to the dissertation problem. Firstly, the machine learning task will be defined, and its experimental evaluation will be discussed. Then, some basic methods for classification and clustering problems will be presented, with a focus on the algorithms utilised in the later research. This will be followed by specific approaches for imbalanced data classification. Next, the concept of MOO *will be described, with examples of the methods, evaluation metrics and solution selection aiding techniques. Lastly, the application of* MOO *in the imbalanced data problem will be researched and analysed.*

2.1 Machine learning task

Machine learning is a part of the Artificial Intelligence discipline aiming to build response models based on given data without explicit knowledge about the underlying processes generating samples [7]. The general form of the machine learning model may be presented as:

$$y = f(x) \tag{2.1}$$

where y is the model's response, and its type varies depending on the nature of the specific problem. $x = [x_1, x_2, \ldots, x_n]^T, x \in X \subseteq \mathbb{R}^n$ is the vector representing an object while x_i are the problems *features*, such as pixel colour, petal's length, network's traffic from the previous day or the result of blood pressure test, and X is called *feature space* [8].

There are different divisions of machine learning tasks depending on, for example, the representation of acquired knowledge or application [9]. However, the most popular one

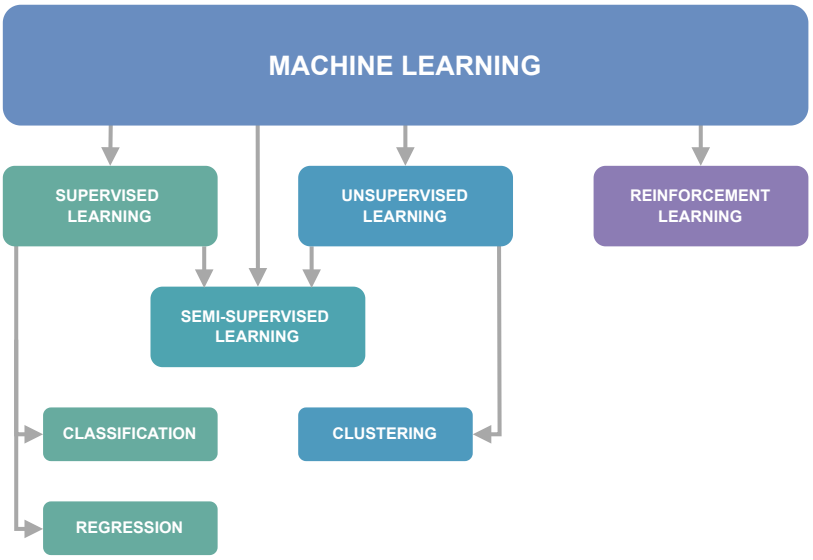


Figure 2.1: *Categories of machine learning tasks based on the labels availability and type*

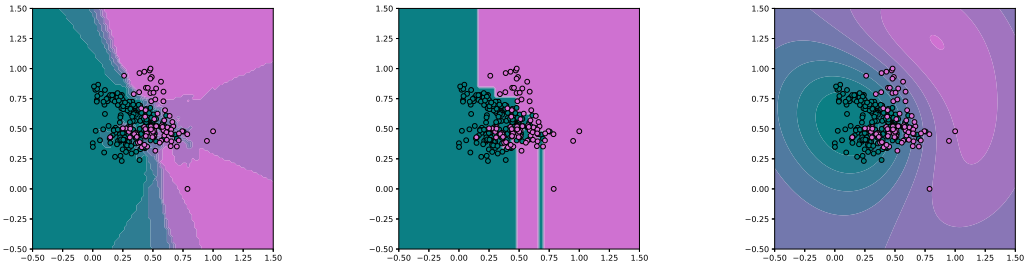
is due to the type and availability of the objects *labels* (Figure 2.1). According to that, we can determine the following categories of machine learning problems [10]:

1. **Supervised learning**, where the model is presented with labels associated with each of the instances and its task is to predict the labels of the unknown samples;
2. **Unsupervised learning**, where the model does not obtain the labels and its task is to, for example, isolate the groups of samples based on their similarity;
3. **Semi-supervised learning**, where only part of the data has assigned labels
4. **Reinforcement learning**, where the model task is to act upon the feedback received from the environment.

Depending on the task type, different models and algorithms are used. In further part of the chapter, categories and models relevant to the thesis will be closely described.

2.1.1 Classification

Classification is one of the tasks of supervised learning where there is a finite number of different labels associated with samples, called *classes*. The classes could be, for example, *healthy*/*diseased* in the case of illness diagnosis or various types of iris flowers. The special case is *binary classification*, when there are only two classes, usually called positive and negative. Any problem with n classes may be transformed into n binary problems [11].



(a) *k Nearest Neighbour* **(b)** *Decision Tree* **(c)** *Support Vector Machine*

Figure 2.2: *Examples of decision regions of different classification models*

A *classifier* aims to map from the feature space to the labels:

$$D : \mathbb{R}^n \to \Omega \tag{2.2}$$

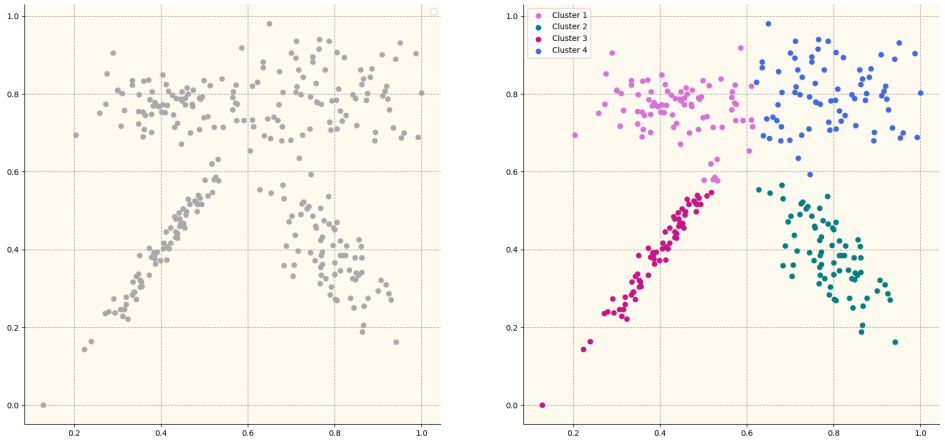
where Ω is the label space. To achieve this, canonical classifiers employ the functions called *the discriminants*:

$$g_i : \mathbb{R}^n \to \mathbb{R}, i = 1, 2, \ldots, c \tag{2.3}$$

which return so-called *support* for each of the classes. The class with the highest support is chosen as the final decision. The feature space can be divided into the *decision regions* based on the discriminant functions scores. The joints of decision regions are called *decision boundaries* [8].

Different classifiers create different decision boundaries even when learning from the same data (Figure 2.2). This is due to differences in assumptions and underlying mathematical models, which constitute each model *bias*, that must be considered when selecting appropriate solutions to the problem.

The models determine the parameters of their discriminants by optimising specific *loss function*, which usually measures the difference between the perfect solution and the current configuration. The most canonical loss function would be *error of misclassification*, which presents the number of incorrectly predicted samples. However, due to the nature of many classification algorithms' parameter optimisation, it is not feasible to use it explicitly [12]. Instead, the so-called *surrogate losses* are employed, which are the functions with properties allowing their inclusion in the model that still contribute to the overall decrease of the classifier's error [13].



(a) *Unlabelled data* **(b)** *Data after clustering*

Figure 2.3: *The example of clustering of unlabelled data*

2.1.2 Clustering

Clustering is one of the tasks of unsupervised learning, meaning that the labels of the samples are unavailable during model training [14]. Its aim is to assign samples to the groups, called *clusters*. The example of clustering is presented in Figure 2.3.

One of the essential concepts of the clustering domain is *similarity*, as there is an assumption that the samples from the same clusters should be as similar as possible, and objects from different clusters should differ significantly. In this context, the similarity of two objects denotes their proximity, and thus, distance metrics are employed as the measure [15]. The most popular and default for many algorithms is Euclidean distance measure, which is a special case of *Minkowski distance* (2.4) when $p = 2$. Another example is *cosine similarity* (2.5) [15].

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{2.4}$$

$$d(x, y) = \frac{x \times y}{\|x\| \|y\|} \tag{2.5}$$

2.2 Experimental design

The basis of the scientific inquiry is a formulation of the hypothesis, which is then verified by experimentation. Although many computer science theories used to be proved by employing mathematics and theoretical debate, nowadays, new knowledge is mainly derived from practical experiments, including the machine learning domain [16]. It is

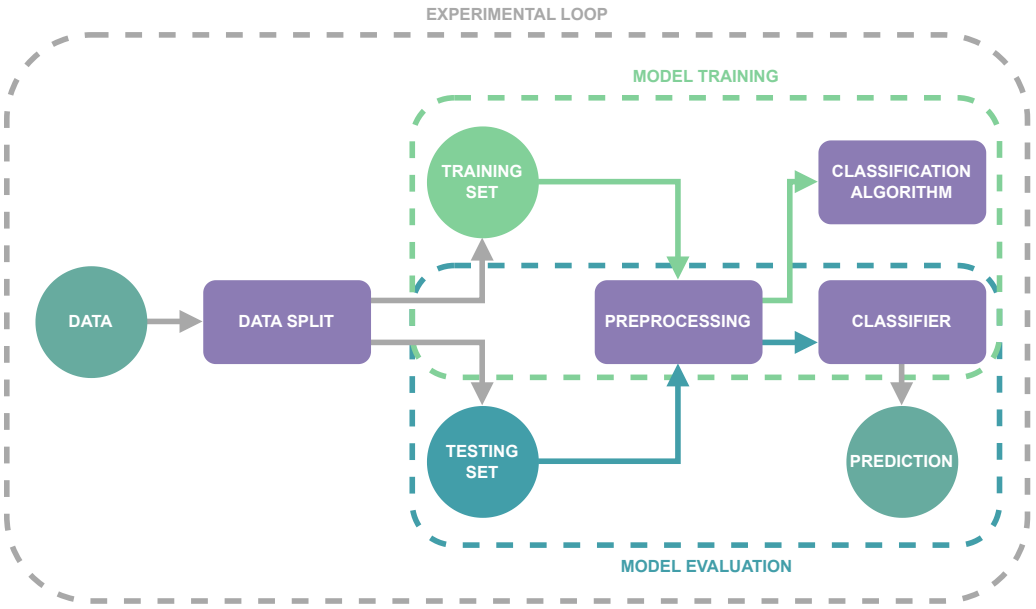


Figure 2.4: *A scheme of the machine learning experimental pipeline*

especially crucial to take appropriate measures while designing an experiment, both to improve the quality of the proposed method and to make its performance more credible and reproducible.

2.2.1 Experimental pipeline

The scheme of the machine learning experiment is presented in Figure 2.4. The experiment consists of two main parts: training the machine learning model and its evaluation, also called post-processing [17]. Both processes should use different subsets of original data obtained by data splitting. Before the data is fed to the algorithm, it needs to be preprocessed. The *preprocessing* is a set of techniques meant to modify original data to improve or even enable classifier training [18]. It includes:

- data cleaning and imputation - as most of the classification models cannot handle not numerical data, some features, for example, categorical, must be transformed into numbers, and missing data need to be filled or, if feasible, discarded;
- feature engineering - consisting of feature selection and reduction- is employed to remove attributes detrimental to the prediction quality or create new features enhancing the classification. It may also lead to a decrease in the cost of data acquisition, especially important in the *cost-sensitive learning* [19], as well as decrease the impact of *the curse of dimensionality*, which hinders models' ability to generalise from sparse data;

- data sampling - crucial for handling noise data and detecting outliers, may improve the generalisation abilities of the model and prevent overfitting. It may also be employed in order to reduce large data sets to optimise computation complexity [18]. Moreover, it is one of the ways of handling data imbalance.

Some processing, like data sampling, can be only performed on data used to train the model, as its employment on evaluation sets might falsify the results. Others, for example, data imputation and attribute generation, are crucial as the classifier is adjusted to the specific feature space.

The whole pipeline is a part of the *experimental loop*. The number of repetitions of the loop, as well as specific data split configuration, depend on the established *evaluation protocol*.

2.2.2 Evaluation protocol

Evaluation protocol determines how to assess classification algorithms so the obtained results are as close to reality as possible. To achieve that, it is crucial to examine how the models predict data not seen before. There are a few different strategies for model evaluation:

1. **Testing on training set** - utilising the same data the model was trained on. It is unacceptable to employ this approach for model assessing and comparison because it does not test the generalisation capabilities of the method, and leads to overoptimistic resuls, as its high quality scores might be an effect of the model remembering the samples instead of learning from them [8]. Nevertheless, many classification algorithms use this method during parameter optimisation, sometimes together with some procedures that prevent overfitting, such as pruning or depth control in decision trees or limiting iterations in neural networks.

2. **Hold-out** is a strategy of splitting data into two disjoint sets of different sizes. The set with more samples is used for training, and the other for model testing [20]. It is the simplest way to ensure that the algorithm is evaluated using samples different from the one it learnt from. However, it is challenging to select the ratio of the training and test sets, as a model trained on more data is better at estimating real distribution, but the less data is used on testing, the less reliable it becomes [20]. Moreover, due to no replication of the split, the quality estimation is less credible, and the model's good performance may be caused by intentional or unintentional *data picking* - selecting the data subsets that fit the algorithm the best. Still, with

the sufficient data volume and ensuring the same distribution in training and test set, the protocol allows precise error estimations [21].

3. **K-fold cross validation** - in this protocol the data is split into k sets of the same size, called *folds*. Each fold is then utilised for testing, while the rest is employed for the classifier training. That gives k iterations of assessing the estimator, which results are then averaged [7]. k usually equals 10 or 30. In general, the smaller the dataset, the bigger k should be chosen, leading even be set to the number of samples so that the test set is degraded to a single object. This approach is called *leave-one-out* [22]. The advantage of the k-fold validation in relation to previous protocols is a decrease in the influence of the split and a better estimation of the real quality of the tested classifier. However, since only a small part ($\frac{1}{k}$) of the whole dataset is used for validation, still the estimation might be inaccurate. Moreover, there is a significant dependency between each of the training sets, as every pair share $\frac{k-2}{k}$ data [7].

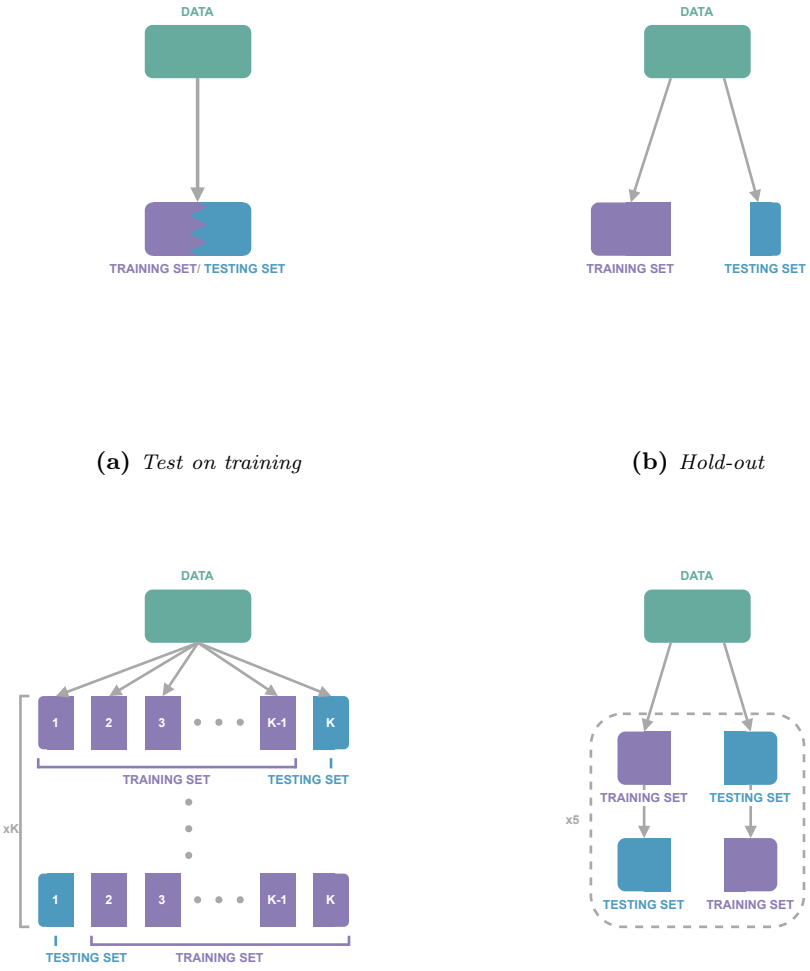
4. **5 x 2 cross validation** - is an improvement of k-fold cross validation [23]. In this strategy, the data is split into two equal sets - the first is employed for training and the other for testing. Then, the second is used for classifier training, and the first is for validation. The whole procedure is repeated 5 times with shuffling the samples in between splits. The value of 5 repetitions is determined experimentally as a trade-off since with too few iterations, the noise in the variance of results has too significant of an impact on the final estimation, and with too many repetitions, there is too big of an influence of the sets dependency [23]. The biggest benefit of this protocol is its quality estimation and the splits in which each sample is used equally for training and testing. However, splitting the data in half can be disadvantageous in the case of small datasets, as there might be too few objects for the model to learn from.

The described protocols are presented in Figure 2.5.

In all of the described protocols, if the data split occurs, it should be random for better quality estimation. However, to better replicate the distribution of the data, especially in a case of imbalanced data, the split should be *stratified*, meaning that the proportion of the problem's classes in the training and test set are the same as in the original dataset.

2.2.3 Results analysis

When experiments are finished, and models' predictions on test data are obtained, their results should be processed and analysed. The post-processing of the experiments can



(a) *Test on training* **(b)** *Hold-out*

(c) *k-fold cross validation* **(d)** *5x2 cross validation*

Figure 2.5: *Visualisations of different experimental protocols*

be divided into two parts - quality measures calculation and statistical analysis.

2.2.3.1 Quality measures

Most of the prediction quality measures for classification are based on the so-called *confusion matrix* (Fig. 2.6) [24]. The confusion matrix summarises how many of the samples were correctly classified. For the binary problems, there are four segments of the table:

1. **True positives** (TP) - number of objects from the positive class that were correctly predicted by the model

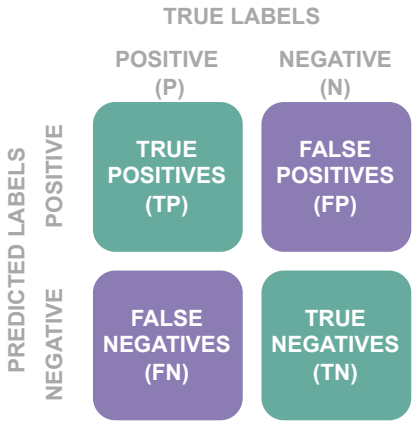


Figure 2.6: *Confusion matrix*

2. **False positives** (FP) - number of objects from the negative class that were predicted by the model as positive
3. **True negatives** (TN) - number of objects from the negative class that were correctly predicted by the model
4. **False negatives** (FN) - number of objects from the positive class that were predicted by the model as negative

The most popular general purpose prediction quality metric is *accuracy* (2.6) or the complementary *error rate* (2.7) [24]. These measures inform how many of the samples were correctly or incorrectly classified compared to the whole dataset.

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

$$error\ rate = \frac{\mathrm{FP} + \mathrm{FN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} \tag{2.7}$$

Both *accuracy* and *error rate* treat every sample prediction with the same importance. However, this assumption is inadequate in the case of class imbalance since, for example, high accuracy values might not indicate good general performance, but disregarding the minority class [25].

One solution for that is the employment of the class-specific measures, such as *recall* (otherwise called *sensitivity* or *true positive ratio* - TPR) (2.8), *specificity* (2.9) or *precision* (2.10). The first two metrics determine what portion of positive and negative classes are correctly classified, whereas precision informs how many of the samples predicted as

positive are actually positive. The main disadvantage of class-specific quality metrics is the fact that they do not indicate the other problem's class prediction, which might cause the model's bias towards the considered class and hinder the analysis of the results and, as a consequence, require presenting various quality assessments [25].

$$recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2.8}$$

$$specificity = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{2.9}$$

$$precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{2.10}$$

Another group of quality measures are *aggregated metrics*, which are the functions of base metrics such as *precision* or *recall*. *balanced accuracy* (BAC) [26] is the average of *recall* and *specificity*:

$$\mathrm{BAC} = \frac{recall + specificity}{2} \tag{2.11}$$

Its main advantage over previous measures is that it considers both classes equally and decreases when one of the classes is predicted poorly.

Gmean (*geometric mean*) score is a similar measure employing both *recall* and *specificity* [27]. However, instead of arithmetic, the geometric average is used:

$$Gmean_1 = \sqrt{recall * specificity} \tag{2.12}$$

To focus more on the correct minority class prediction, another version of *Gmean* was also proposed [28], employing *recall* and *precision*:

$$Gmean_2 = \sqrt{recall * precision} \tag{2.13}$$

Another widely popular imbalanced data quality measure is F_β *score*, which is the harmonic mean of *precision* and *recall*:

$$F_\beta score = \left(1 + \beta^2\right) \frac{precision * recall}{\beta^2 precision + recall} \tag{2.14}$$

β parameter is a weighting factor between *precision* and *recall* and should be higher when predicting all of the minority samples is more important and lower if its correct classification proves significant. Usually, both factors are treated equally and $\beta = 1$. However, some studies claimed that it should be selected more thoroughly, for example, by being dependent on minority and majority classes ratio, to decrease the bias towards majority class [29].

Another way of classifier evaluation is by employing the ROC - *receiver operating characteristic* curve, showing the relationship between *true positive rate* (TPR) and *false positive rate*. The curve represents one classifier with different parameter values [17] or different prediction thresholds [30]. If only one version of the model is considered, then ROC has the form of a single point. The measure encapsulating the ROC curve is *area under* ROC *curve* (AUC), which, as the name suggests, is obtained by calculating the area between the x-axis and the ROC curve with one as a limit. In the case of binary classification and single point 'curves', the AUC measure is the same as the BAC score.

Several other classification metrics were proposed, such as different variants of AUC [31–33], *Index of Balanced Accuracy* [34], *Matthews Correlation Coefficient* [35], *Cohen's κ-measure* [36] and many more.

Studies have shown that even similar evaluation metrics differ in their results and should not be easily omitted, as very often the differences between compared methods are very tight, and the best model might vary depending on the selected measure [37]. Nevertheless, to avoid overly complicating the analysis of the evaluation results, the number of presented quality measures should be reasonable [25].

2.2.3.2 Statistical evaluation

Once the experiments' results are obtained and measures calculated, the statistical evaluation must be conducted since the difference between each classifier scores might not be significant and stem not from the model's better quality but from specific data sampling or the method's randomness [38]. The selection of statistical tests depends on the numbers of compared classifiers, datasets and the distribution of the results.

The first group of statistical tests is dedicated to comparing two models on one dataset. One such method is *McNemars test*, which examines whether the difference between unique mistakes made by both classifiers is significant [23]. The advantage of McNemar's test is its low type 1 error (when the significance in the difference in classifiers' performance is accepted, while in truth, there are none). However, it can be conducted only for one run, meaning that it has to be assumed that there is little variance between employed and real data [23].

As hold-out should not be employed if possible, other tests must be considered. *Paired t-test with correction* [39] could be used in a case of k-fold cross validation and *combined 5x2 F-test* [40] for 5x2 cross-validation protocol. Both tests check whether the mean of the differences of the classifier's prediction quality on all of the folds is close to 0, employing appropriate statistics which consider each protocol's assumptions, for example, the fact that the data samples in each fold are not independent.

The tests above cannot be employed to make a more general comparison over multiple datasets, as the differences in quality on different data are not commensurable (as they do not come from the same distribution) [41]. One of the tests dedicated to comparing two methods on several problems is *Wilcoxon sign-rank test* [42]. Instead of calculating statistics based directly on quality disparity values, the test ranks the differences and employs their sums. Another way of statistical evaluation in such a scenario is *signed test* [43], which only counts wins of every classifier and is based on the intuition that when of similar quality, each of the models should win approximately half the time. Neither of the tests assumes the normality of the distribution of the results, which, on the one hand, makes them more applicable, but on the other, weaker than different *parametric* tests, such as the paired t-test [41]. As the Wilcoxon test takes into consideration not only how many times but also with what magnitude the model won, it is stronger and shows the significance of the results more often than the signed test [41].

The last group of tests is dedicated to comparing more than two classifiers. Such evaluation consists of two parts: firstly, the test is conducted to examine whether there is a significant difference in multiple models' performance, then a post-hoc test is employed to determine which of the compared algorithms is better than the others [41]. One of the most popular tests for comparing multiple models is ANOVA (*analysis of variance*) [44], with Tukey post-hoc test [45]. The main disadvantage of the ANOVA is that it assumes the normality and sphericity of the data distribution, which is not granted in the case of classifiers results [41]. Friedman test [46] is nonparametric equivalent to the ANOVA. It ranks performances for every problem and then calculates the statistic based on the average rank of each model. If the determined statistic proves the significance of the results, the *Nemenyi post-hoc test* can be employed, which calculates *critical difference* based on a number of compared algorithms and evaluated datasets. The critical difference determines how much the average rank of the model must differ to verify statistical dominance.

Statistical evaluation is essential for the credibility of the experiments' results. However, it is crucial to select and analyse the results of the statistical test correctly. In literature, there are occurrences of improper statistical evaluation, such as using multiple pairwise tests to compare several versions of the models [43] or employing parametric assessments without determining their assumptions fulfilment [41]. Moreover, the results of statistical tests might be manipulated by adequate selection of the data splits or pool of compared algorithms [38].

It must be noted that there are some critics to the established statistical evaluation methodology, both with the methods used and drawn conclusions. They lead to proposition of different approaches, such as Bayesian analysis [47].

2.3 Chosen machine learning algorithms

2.3.1 Naive Bayes Classifier

The Naive Bayes Classifier is an example of a probabilistic estimation model. It uses *a posteriori* class probabilities to select the class with the smallest risk of wrong prediction. The term *naive* comes from the assumption that each of the problem's features is conditionally independent [8]. The formula for choosing the label is as follows:

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i \mid y) \tag{2.15}$$

where $P(y)$are the *a priori* probabilities of the classes and x_i are the consecutive features. Different variants of Naive Bayes models use distinct assumptions about features' distribution; for example, the Gaussian Naive Bayes classifier assumes that the distributions are normal.

2.3.2 K Nearest Neighbour Classifier

K Nearest Neighbour Classifier (KNN) is an example of a density estimator, aiming to predict samples' classes without any knowledge about their distributions. It assumes that instances with the same labels are somehow similar to each other and lay in close proximity in the feature space [48]. When predicting the label of the samples, the model searches for k instances from earlier obtained data closest to the examined point. The label that belongs to the majority of such samples is selected as the prediction (Figure 2.7). KNN model is often called *lazy learner*, as during training, it does not optimise any discriminant function but checks the data during the prediction. The only parameters of the KNN model are *the distance metric* and k value. The most popular distance measure is Euclidean metric 2.4. The appropriate selection of the value of nearest samples is crucial and depends on the problem characteristic, such as the number of instances, classes or their disproportion. Usually, k is chosen to be odd, for example, 3 or 5, to prevent draws.

2.3.3 Decision Tree Classifier

Decision Tree Classifier is a sequential model consisting of a set of attribute values tests. Each test, usually checking a single feature, creates a branch leading to a node with either another decision rule or the final response (class label) called a *leaf* [49]. The example of the decision process is presented in Figure 2.8.

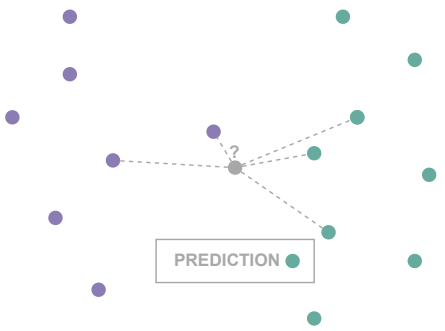


Figure 2.7: *Example of* KNN *decision process with* $k = 5$

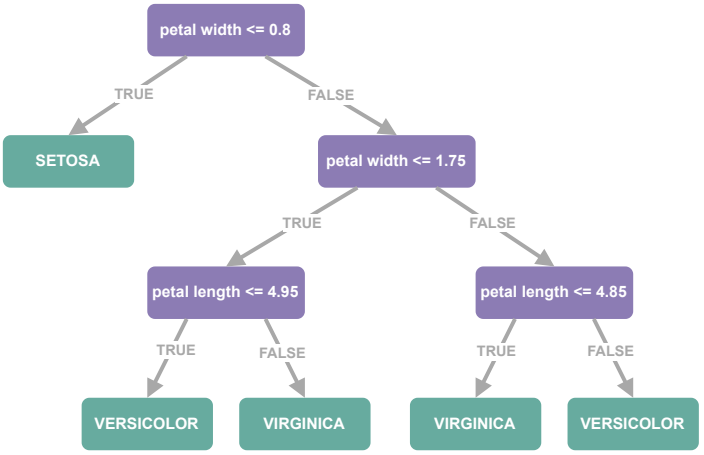


Figure 2.8: *Visualisation of the decision tree trained on* iris *dataset*

During training, each node test is selected based on the *split criterion* - each potential feature space split is evaluated, and the one with the best score is executed. There are several split measures, for example, Information Gain (2.16) or Gini impurity (2.18) [50].

$$InformationGain(x_i, D) = Entropy(y, D) - \sum_{v \in x_i} P_v Entropy(y, D_v) \tag{2.16}$$

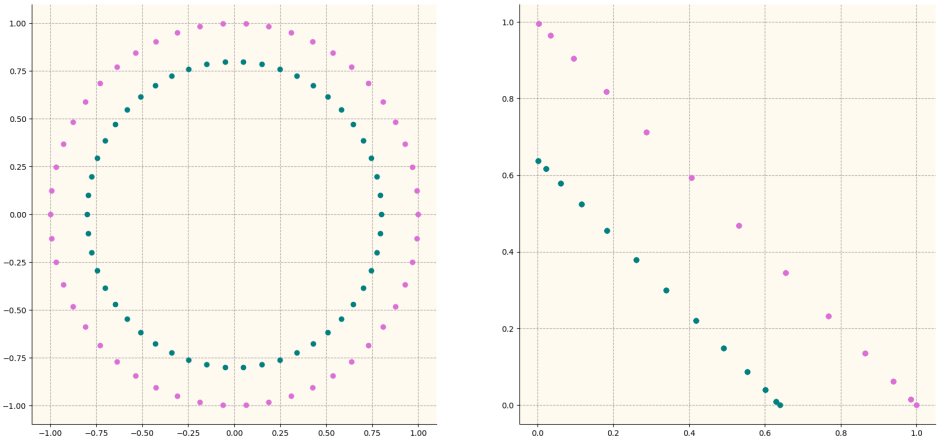
where x_i is examined feature, v are different values of x_i, P_v is a relative frequency of value v, and

$$Entropy(y, D) = \sum_{c \in y} -p_c \log_2 p_c \tag{2.17}$$

while

$$Gini(D) = 1 - \sum_{i=1}^{C} P_i^2 \tag{2.18}$$

where P_i is a relative frequency of the class i in the examined node.



(a) *Original space* **(b)** *Transformed space*

Figure 2.9: *Example of transformation* $\varphi(x_1, x_2) = (x_1 * x_1, x_2 * x_2)$

The biggest advantages of decision trees are their interpretability [49] and flexibility, as they are part of more complex and acclaimed algorithms such as Random Forest [51] or XGBoost [52].

2.3.4 Support Vector Machines

Support Vector Machines (SVM) [53] aim is to find the hyperplane dividing the problem's classes. Since many problems are not linearly separable, to achieve this, the algorithm converts the samples to other- usually higher dimensional- space, where such partition is possible, using some mapping $\varphi(x)$ (Figure 2.9). To reduce calculation, the SVMs employ so called *kernel trick*, when instead using function φ explicitly, they use *kernel function* K that equals the dot product of two vectors [54]

$$K(x_a, x_b) = \varphi(x_a) \cdot \varphi(x_b). \tag{2.19}$$

Some examples of popular kernels are polynomial (2.20) or Gaussian RBF (2.21)

$$K(x_a, x_b) = (x_a \cdot x_b + r)^p \tag{2.20}$$

$$K(x_a, x_b) = \exp\left(-\frac{\|x_a - x_b\|^2}{2\sigma^2}\right) \tag{2.21}$$

where p is a chosen polynomial degree, r is added bias and σ is tuning parameter.

2.3.5 Artificial Neuron Networks

Artificial Neural Networks (ANN) are models designed to mimic the way of working of the human brain [55]. They are composed of the processing units called *neurons* (Figure 2.10), which consist of:

- input vector $u = [u_1, u_2, \ldots, u_q]$
- weights vector $w = [w_1, w_2, \ldots, w_q]$
- a bias b
- activation function ϕ

The result output v of the neuron has a form [8]:

$$v = \phi\left(\sum_{i=1}^{q} w_i u_i + b\right) \tag{2.22}$$

Multiple activation functions where proposed, for example heaviside step 2.23 or rectified

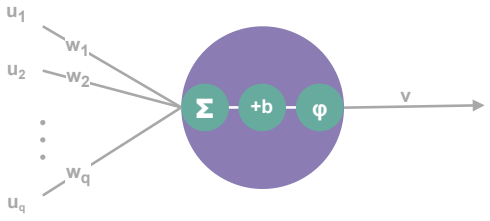


Figure 2.10: *Neuron unit scheme*

linear unit (relu) 2.24 function:

$$\phi(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases} \tag{2.23}$$

$$\phi(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} \tag{2.24}$$

Neuron processing units are connected in a structured way, aligned in connected *layers*. One of the easiest and most popular neural network models is Multilayer Perceptron (MLP) (Figure 2.11). The discriminant function of the MLP model depends on the number of layers and its activation, partitioning data into two planes with one layer and creating more complex regions the deeper it gets [56]. Another example of models is convolutional

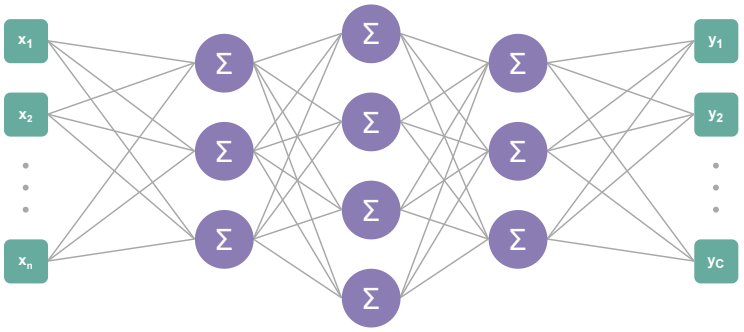


Figure 2.11: *Example of multilayer perceptron*

neural networks (CNN) for an image or sequential data [57] or transformers for natural language processing [58].

During training, the weights of the neuron connections are usually set at random and then optimised based on the intended output using a technique called *back propagation*, where the difference between obtained and true predictions influence each of the layers sequentially [56]. There are several ways of optimising network parameters, but *gradient* methods are the most popular. One of them is *stochastic gradient descent* calculating the weights as follows [59] :

$$w_{t+1} = w_t - \alpha_t C \frac{\delta l(x_t, y_t; w_t)}{\delta w} \tag{2.25}$$

where $l(x_t, y_t; w_t)$ is selected *loss function* which we want to minimise in the model, α_t is a *learning rate* and can be time dependent and C is positive-definite matrix. A more advanced neural network optimiser is *Adam* (adaptive moment estimation) [60], which, apart from explicitly using loss function gradient, also utilises its momentums.

As gradients are used to optimise neural network parameters, it is crucial that a loss function is differentiable.

2.3.6 Ensemble learning

Ensembles of classifiers are the models which combine several estimators and aggregate their predictions. Classifier committees have their foundations, among others, in Francis Galton's concept of *wisdom of the crowd* [61] and Cordocet's *jury theorem* [62]. The latter proves that in a situation when there is a group of independent estimators of equal quality (although better than random) aggregated by selecting the response stated by the majority:

1. The overall committee quality is better than the quality of a single member;

2. The quality increases with the increase in committee size.

There are, however, some counterarguments that independence between classifiers is not a sufficient condition and that it is error committing between models that must be independent [63].

This fact, together with modularity and the ability to decompose the problem, resulting in more complex decision boundaries, lead to the good quality and popularity of the ensemble methods [64].

An ensemble model consists of two components: *a pool of estimators* and *a combiner* aggregating their predictions (Figure 2.12) [8]. One of the most important aspects of the selection and training of ensemble members is their *diversity* [65], as similar classifiers tend to make the same mistakes - thus, there is no benefit in their combination. There are different ways of assuring the diversity of estimators, and they can be assigned into three categories [64]:

1. diversity by data alteration - meaning that each of the classifiers is trained using different data obtained from the original set; it can be done for example by selecting different subsets using *bootstrapping*. Another strategy is to train classifiers using different problem features [65]

2. diversity in classification algorithms - as different classifiers have different biases, combining them may lead to more complex decision regions and quality improvement. The diversity might be in general classification models (e.g., KNN, SVM, etc.), in that case, an ensemble is called *heterogeneous*, or in algorithms parameters (e.g., different kernel functions in support vector machines or learning rates in neural networks) [8].

3. diversity by output manipulation - refers mostly to multi-class problem decomposition by, for example, binarisation with techniques such as *one-vs-one*, *one-vs-all* or *Error-Correcting Output Codes* (ECOC) [66], where each of the ensemble members trains from a different subproblem (i.e. two of the classes distinction) and their output is later aggregated in a way to match the main problem's classes.

Depending on the chosen combiner and specific problem, it may be necessary to conduct a *pruning* procedure to reduce the ensemble size. It is especially important when training members on subspaces of data (both vertical and orthogonal), as some classifiers' contribution might be detrimental to the overall committee's quality [67]. One of the examples of pruning is by choosing the n best classifiers or by greed search [8]. Another example is to use some kind of metaheuristic, such as a genetic algorithm [68].

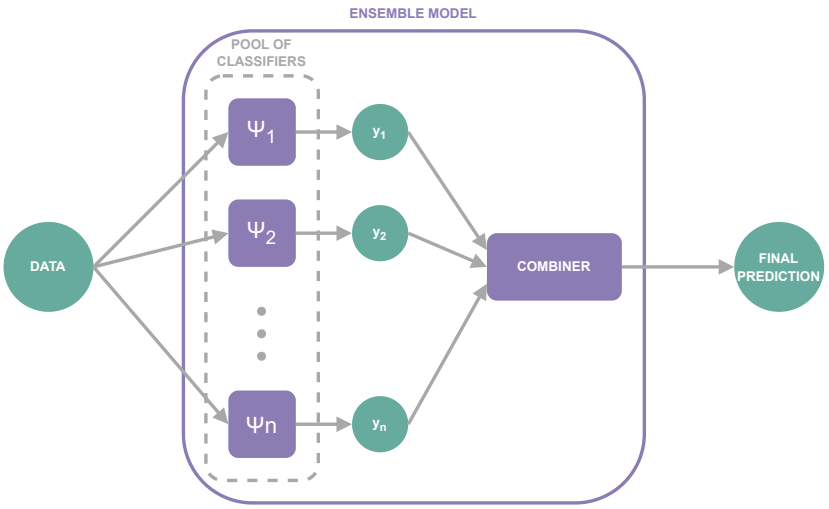


Figure 2.12: *Scheme of an ensemble classifier*

The other crucial aspect of ensemble learning is the way of aggregating members' predictions. The most basic way of combining classifiers is via *the majority voting*, where the response given by the biggest number of estimators is chosen as the final response [8]. A slightly more complex and popular one is *weighted majority voting*, where each classifier is assigned a weight, influencing their contribution in the final prediction. The response from the model using this type of combiner is as follows [17]:

$$\Psi(x) = \arg\max_{j \in \mathcal{M}} \sum_{i=1}^{n} [\Psi_k(x) = j] w_i \tag{2.26}$$

where w_i is the weight of i-th ensemble member and [] stands for Inverson's bracket:

$$[x] = \begin{cases} 1 & \text{when } x = true \\ 0 & \text{when } x = false \end{cases} \tag{2.27}$$

There are a few different ways to assign weights to the classifier [17], among others:

- each classifier has one weight assigned, for example, proportionate to its accuracy [8]
- weights are assigned to each classifier for each class, for example, based on its recall [69]
- as above, but additionally weights depend on x [17]

Another way of aggregating members' decisions is to utilise meta-learning methods - classifiers trained on the other estimators' prediction [64]. In this case, the training

set has the same number of samples as the original set used to train the committee members, but the number of features depends on the ensemble size. Various classification algorithms could be employed, for example, Naive Bayes classifier [8] or MLP [70].

Instead of basing the response on labels returned by base classifiers, estimators' confidence can also be taken into account, for example, in the form of each class's support [71].

There are several ensemble models and strategies. One of the most popular to employ or improve are:

- Bagging ensemble (**B**ootstrap **agg**regat**ing**) [72] is an ensemble method utilising bootstrapping to select data for each member training. The samples are drawn from the original set with replacement, meaning that each object may appear more than once. Members' predictions are aggregated via majority voting. This technique is especially prominent while employing unstable classification algorithms (meaning that a slight change in data creates a significant change in the decision boundaries). such as Decision Trees;
- AdaBoost (**Ada**ptive **Boost**ing) [73] is an ensemble algorithm employing *boosting* strategy, meant to improve each iteration of the model. In this algorithm, each of the data samples is assigned a weight, which is sequentially increased or decreased based on its difficulty, calculated as the examined estimator's exponential loss. This technique ensures that as the generation of ensemble members proceeds, there is a bigger focus on the samples that prove difficult to predict. The crucial assumption of the algorithm is that it utilises *weak learners*, meaning that at the base, their quality is slightly better than random and that their premised decision boundaries are not too complex so that there is the possibility to improve their quality and to prevent overfitting. Weighted majority voting is used to aggregate the members' predictions. Instead of adding new estimators one by one in greedy approach, the whole ensemble can be optimised using gradient, just like in XGBoost model [52]
- Random Forest [51] is an ensemble consisting of Decision Trees. Each tree is created using a bagging procedure. Additionally, to introduce even more randomness, the subset of random features is drawn for each split, and the best choice of attribute is selected from that group, instead of from the whole feature space. Moreover, the trees created utilising this technique are not pruned. Decisions of the members of the ensemble are then aggregated using majority voting. One of the most significant advantages of Random Forest ensemble is its generalisation abilities, even without parameters tuning, and its robustness against noisy samples [74].

2.3.7 Clustering algorithms

There are many approaches to the clustering problem. One of the categories of clustering methods is algorithms based on the partition, which aim to divide the feature space by finding the centres of the clusters [75]. One of the most popular examples is K-means algorithm[76]. Its goal is to find k centroids that would minimise the distances to cluster members. The scheme of the K-means algorithm is as follows:

1. Initialise clusters centers $c_1, c_2, \ldots, c_k$
2. Assign samples to clusters by finding for each x_i

$$\underset{j \in \{1,2,\ldots,k\}}{\arg\min} \; d(x_i, c_j)$$

3. For each cluster $C_1, C_2, \ldots, C_k$ update its center

$$c_i = \frac{1}{\|C_i\|} \sum_{x \in C_i} x$$

4. Calculate the error E

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} |x - c_i|^2$$

5. Repeat until E does not change significantly or cluster memberships no longer changes

The biggest disadvantage of the partitioning algorithms is their dependence on the k parameter, which has to be selected based on expert knowledge, problem constraints or extensive experimentation. Multiple measures were proposed to determine the best number of clusters [77]. Nonetheless, they tend to favour the least or the most groups possible. Another weakness of K-means and other centroid-based methods is that they are not appropriate for non-convex data.

Another clustering strategy is to group samples based on their density [75]. Methods of this type make an assumption that data in the dense areas belongs to the same cluster. One such algorithm is DBSCAN [78], which connects points based on their density-reachability, that is, their appropriate proximity (lesser than given ϵ) and being surrounded by different objects. OPTICS [79] is an expansion to DBSCAN, computing the clusters for different values of the neighbourhood size. Both of the methods assume the existence of noise - samples not belonging to any of the clusters. Moreover, selecting the value ϵ is crucial and has the biggest influence on the created clusters and determining noise objects.

2.4 Methods for imbalanced data

The problem of data imbalance poses a significant challenge since it affects correct recognition of the quintessential samples, such as occurrences of fraud or network attacks, which very often are not as numerous as less important objects. There were proposed multiple approaches to countering class imbalance in data, which can be grouped into three categories [80]:

- *data-level methods*, which aim to balance classes ratio by adding or removing samples;
- *algorithm-level methods*, that change pattern recognition algorithm to take all of the classes into consideration;
- *hybrid methods*, which combine both of the above, often in the form of classifiers ensembles.

2.4.1 Data-level approaches

The first group focuses on modifying the distribution of the classes by, for example, removing or generating new samples, which can be later used to train a classifier not adjusted to the skewed data. The most basic methods are random under- and oversampler, which randomly remove majority class samples or duplicate minority class samples. They are computationally low cost, however they might lead to the erasure of the crucial information or enhancement of the noise [80], which is the reason why most of the sampling methods employ more guided techniques.

One of the most popular oversampling algorithms is *Synthetic Minority Over-Sampling Technique* (SMOTE), which generates new objects by interpolating selected minority samples and their randomly chosen nearest neighbours [81]. The algorithm allows the broadening of the minority class decision regions and better generalisation. However, it may also result in amplifying the influence of noise samples or even creating new ones. Many algorithms based on SMOTE were proposed to toggle its problem with creating unnecessary detrimental samples [82]. *Borderline SMOTE* focuses on creating samples around borders of minority class clusters so that the decision regions are better defined and potentially noise objects ignored [83]. On the other hand, *Adaptive Synthetic Sampling Approach* (ADASYN) creates more samples around minority class objects that may be harder to predict because of their surroundings [84]. Other approaches include generating new samples and removing the ones which are detrimental to the prediction [85, 86], modifying how the points are interpolated [87, 88], or employing SMOTE in the clusters

of samples determined for example by K-Means algorithm [89, 90]. There were also proposed methods, such as DeepSMOTE [91], incorporating SMOTE-like points generation into deep learning models.

Another method for data preprocessing is undersampling, which removes majority class samples. Many early algorithms, such as *Condensed Nearest Neighbour* (CNN) [92], *Edited Nearest Neighbour* (ENN) [93] or *Tomek Links* [94], were based on nearest neighbour approach. CNN removes samples that do not change the decision of the classifier model. ENN removes samples in which the majority of neighbours are from different classes. The last algorithm removes so-called Tomek links, that is, objects with the nearest neighbour from another class. In the case of imbalanced data, each method is conducted on samples from the majority class. The problem with these methods is that they do not guarantee the balancing of the classes and may not improve the recognition of the minority class, especially in the case of a significant disproportion between classes. Clustering-based undersampling divides majority class samples into clusters and leaves only one representative from each group [95], while *Radial-Based Undersampling* removes majority class samples with the biggest mutual-class potential [96] or *Evolutionary Undersampling* which employs evolutionary algorithm to select optimal subset of the training set [97]. The biggest disadvantage of undersampling techniques is that they decrease the size of training sets, which can be detrimental to the predictive abilities of the classifiers.

Hybrid sampling algorithms combine both over and undersampling, improving minority class recognition by both creating new examples and cleaning the areas of interest. One such technique is a *combine cleaning and resampling algorithm* (CCR) and its radial-based modification (RB-CCR), which cleans majority class in the areas around minority class samples, with the possible variant of deleting them, and populate it with new synthetic points [98, 99]. Different approaches examples are to employ SVM to delete majority class samples far from the decision boundary and then perform SMOTE on the groups of the remaining set [100] or dividing training data into clusters and performing over- or undersampling based on the number of minority class samples [101].

Figure 2.13 presents the examples of results of different over- and undersampling algorithms.

2.4.2 Algorithm-level approaches

Another way to deal with imbalanced data problem is to adapt the classification algorithms so that they do not ignore the disproportion between samples[80]. One such approach is *cost-sensitive learning*, which takes into consideration the cost coming from

(a) *SMOTE* **(b)** *Borderline SMOTE* **(c)** ADASYN

(d) *k-means SMOTE* **(e)** *SMOTE with Tomek Links* **(f)** *CCR*

(g) *CNN* **(h)** *ENN* **(i)** *Tomek Links*

Figure 2.13: *Examples of different sampling algorithms processing*

the misclassification of the objects or from data acquisition [102]. An example of accommodating differences in class importance is to assign higher weights and misclassification costs to the minority class samples in models like KNNs [103], SVMs [104, 105] or Naive Bayes classifier [106]. In the case of neural networks, the most straightforward and popular algorithm-level approach is to employ a loss function that differentiates class influences [107]. Several optimisation objectives were proposed, such as *Mean False Error* [108], which calculates the mean error of each class prediction separately:

$$FPE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.28}$$

$$FNE = \frac{1}{P}\sum_{i=1}^{P}(y_i - \hat{y}_i)^2 \tag{2.29}$$

$$MFE = FPE + FNE \tag{2.30}$$

Where FPE and FNE are false positive and negative errors, respectively, N and P are a number of negative (majority) and positive (minority) samples, y_i is i-th output of the model while $\hat{y}_i$ is i-th true value. *Mean Squared False Error* is the improvement of MFE, which utilises both false positive and negative errors; however, it aggregates them in a different manner so that it is more sensitive to the minority class recognition mistakes [108]:

$$MSFE = FPE^2 + FNE^2 \tag{2.31}$$

Another approach would be to employ *cross-entropy* loss function with appropriate class weights [109]:

$$CE = \sum_{i=1}^{C} -w_i[y = y_i]\log p_i \tag{2.32}$$

Where w_i is the weight of the i-th class and p_i is the model's support for the i-th class. The weights should be normalised, either before assignment or during samples' loss aggregation.

The other proposed loss function is focal loss [110], which is a different modification of basic cross-entropy but with a modulating factor that, apart from compensating for class imbalance, also forces the model to focus on harder examples:

$$FL = \sum_{i=1}^{C} -y_i(1 - p_i)^{\gamma}\log p_i \tag{2.33}$$

Where γ is the tuning parameter.

The main challenge of cost-sensitive learning and weight assignment is determining the costs of misclassification in cases when it is not given by experts. Common approaches are setting weights proportional to the ratio of class sizes [109] or optimising them based on the performance objective [103].

2.4.3 Hybrid approaches

Finally, the last methodology is a hybrid approach combining sampling and algorithm adjustment. The common method to merge the two techniques is in the form of an ensemble of classifiers, where sampling is used to provide diversity in data utilised in each member training, and their predictions are aggregated in a way fitting imbalance problem [111]. The bagging technique is altered to better suit skewed class ratio by changing the way each of the subsets is drawn from the original data by, for example, always choosing all of the minority samples and only bootstrapping from majority class [112], drawing from both of the classes separately to obtain subsets which are "roughly balanced" [113], differentiate the probability of being drawn for every sample based on its type and neighbours [114], or employ sampling methods, such as SMOTE, either during bootstrapping [115] or after to balance individual subsets [114, 116]. Boosting algorithms are also enhanced by adding techniques to deal with data imbalance, for example, in the form of different samplers such as SMOTE [117], Random Undersampling [118] or Evolutionary Undersampling [119]. Another proposition is to assign bigger weights for boosting the minority class samples, especially at the borders of classes [120]. Ensemble learning is also a good approach to decrease the risk arising from undesirable sampling since data can be split and processed multiple times, separately for every committee member. For example, [121] proposes dividing majority class samples randomly into bins and combining each group with the whole minority class, with weighting each of the resulting classifiers based on the predicted object's placement in feature space. On the other hand, in [122] ensemble is built on cost-sensitive decision trees, which are trained on different subspaces of features and pruned based on an evolutionary optimisation algorithm, while [123] constructs and ensemble from cost-sensitive neural networks, which differ in the cost assign to both of the classes. Lastly, some problems are naturally modular, such as stream learning or multiclass classification, thus ensembles employment is widely popular. For imbalanced data streams, each chunk of data might be preprocessed to balance the classes by using different sampling algorithms [124] or by aggregating historic objects [125, 126]. Resulting estimators are then weighted based on measures appropriate for class disproportion, such as BAC [124] or Hellinger's distance [127]. In the case of classification with multiple classes, one solution could be to decompose the problem into smaller ones, for example, recognising one class [128] or differentiating between two classes [129], where imbalance would have less of the impact on the final prediction.

2.5 Multi-objective optimisation

The goal of *multi-objective optimisation* (MOO), as opposed to traditional single-objective optimisation (SOO), is to optimise more than one criterion at the same time [130]. Formally, for the problem with m objective functions $f_1 : \mathcal{X} \to \mathbb{R}, \ldots, f_m : \mathcal{X} \to \mathbb{R}$ with $\mathcal{X}$ as solution space, where $m > 1$, the aim of optimisation algorithm is to

$$\text{minimise } f(x) = (f_1(x), f_2(x), \ldots, f_m(x)),\ x \in \mathcal{X} \tag{2.34}$$

The simplest, but also popular, approach is to reduced the problem to single-objective optimisation by aggregating the criteria into one function, although it does not guarantee obtaining the optimal value for any of the factors [131]. Very often, the criteria somehow contradict each other, for example, the price and quality, and it is impossible to find a solution optimal for every objective simultaneously as well as determine which solution is better than the rest. For this reason in a multi-objective environment, two solutions are compared in the context of *Pareto dominance* [132], which is defined as follows: given two vectors $u = [u_1, \ldots, u_m] = [f_1(x_1), \ldots, f_m(x_1)]$ and $v = [v_1, \ldots, v_m] = [f_1(x_2), \ldots, f_m(x_2)]$ v is Pareto dominated by u (denoted $u \prec_{pareto} v$) if and only if

$$\forall i \in \{1, \ldots, m\} : u_i \leq v_i \ \wedge \exists i \in \{1, \ldots, m\} : u_i < v_i \tag{2.35}$$

This means that all criteria values have to be at least equal, and at least one criterion has to be lesser in order to dominate over the other optimisation solution [133]. A solution is *Pareto optimal* when it is not dominated by any other vector:

$$\exists! x \in \mathcal{X} : x \prec_{pareto} x^* \tag{2.36}$$

A *Pareto optimal set* consists of all solutions which are Pareto optimal, while a *Pareto front* contains their placement in objective space [133]. The example of Pareto dominance is presented in Figure 2.14.

2.5.1 Evolutionary optimisation algorithms

One group of multi-objective optimisation algorithms is population-based *evolutionary algorithms* [130]. Just like their single objective counterpart, they consist of a group of individuals representing the solutions to the problem and several sequentially executed operators:

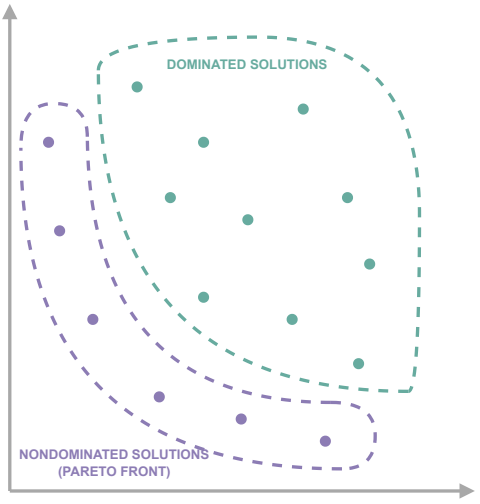


Figure 2.14: *Illustration of the Pareto dominance for minimisation problem*

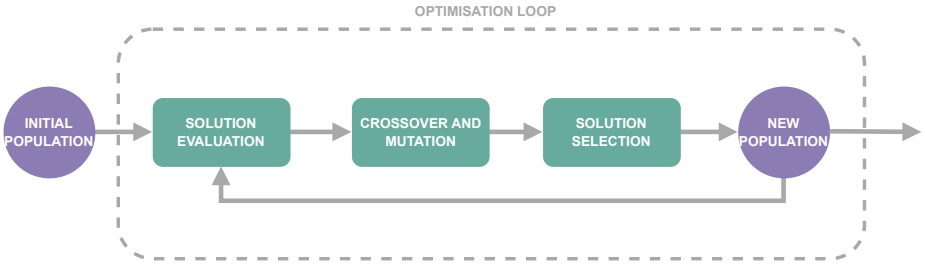


Figure 2.15: *Diagram of a standard genetic algorithm*

- selection, which ranks the individuals based on given objectives and chooses the ones which will be employed in the next iterations;
- crossover, which combines the best solutions in order to create new ones;
- mutation, which modifies the individual solution in a randomised way to possibly generate features not existing in the population.

The steps of the algorithm is shown in Figure 2.15.

The crossover and mutation are *variation* operators dedicated to creating new individuals. While crossover merges two of the best solutions to obtain even better results, the mutation is meant to prevent converging into local optima. Both of these operators are usually independent of the optimisation algorithms and are determined by the problem. The distinction between evolutionary algorithms (single and multi-objective) comes mostly with different selection operators since solutions are assessed and removed or preserved in the population [130].

One of the most popular MOO evolutionary algorithms is *Nondominated Sorting Genetic Algorithm* (NSGA) II [134]. The individuals are selected based on their *nondomination rank*, where nondominated solutions are given the highest first rank and the rank of the remaining points is based on the lowest rank of the solutions the particular individual is dominated by (meaning the rank 2 goes to the points dominated by rank 1 etc.). Additionally, to ensure more diversity *the crowding operator* is introduced, which, in the case of two points of the same nondomination rank, selects the point with a bigger distance to adjacent solutions. The NSGA II also contains elitism, meaning that the solutions from previous iterations remain in the population pool, which, on the one hand, prevents the loss of possibly the best solutions and accelerates the convergence, but on the other hand, may also result in algorithm getting stuck in local optima, so appropriate mutation mechanisms need to be introduced [135]. There are multiple advantages of utilising NSGA II, such as its low computational complexity and lack of algorithm-specific parameters [136].

Another example of a Pareto-based MOO algorithm is the Strength Pareto Evolutionary Algorithm (SPEA) 2, which assigns fitness based on the number of dominated solutions and density in proximity to the point [137]. Other approaches include indicator-based algorithms, such as SMS-EMOA [138], which aim to optimise specific measures of Pareto front assessment, for example hypervolume, and decomposition-based techniques, like NSGA III [139] or MOAE/D [140], that divide main problem to subproblems utilising scalarisation, which are then locally optimised. However, most of them are either computationally complex or require prior knowledge about search space [130].

2.5.2 Gradient algorithms

While evolutionary optimisation algorithms are popular thanks to their versatility and ease of customisation to the problem, they are also time-consuming due to the necessity of performing objective calculations many times (depending on the population size and number of iterations) and, to some degree, random solution searching. Because of this, together with the expanding utilisation of deep learning models, there is a development of gradient-based methods utilising multi-objective optimisation [141].

One of the most straightforward MOO gradient methods is *Multiple-gradient descent algorithm* (MGDA) [142]. It is based on the premise of Pareto stationarity, which is defined as a point where a linear combination of the gradients equals zero, and the selection of descent direction according to gradient vectors of all objective functions. To obtain a set of Pareto optimal solutions, it is required to run the algorithm multiple times with different starting points.

One of the most significant drawbacks of the MGDA method is that it does not ensure a generation of the wide Pareto front, as the obtained solutions only depend on the starting point, and the trade-offs between objectives cannot be controlled. One of the answers to that problem was proposed in the *Pareto Multi-Tasking Learning* algorithm (PARETOMTL) [143]. To ensure the diversity between solutions, objective space is divided into subspaces based on provided preference vectors. Each of the created subproblems is then solved until the Pareto critical point (when there is no possibility of obtaining better objective values without impairing the others) is accomplished. Each subproblem is independent of the other, allowing for parallelisation of the calculations, thereby shortening the training time.

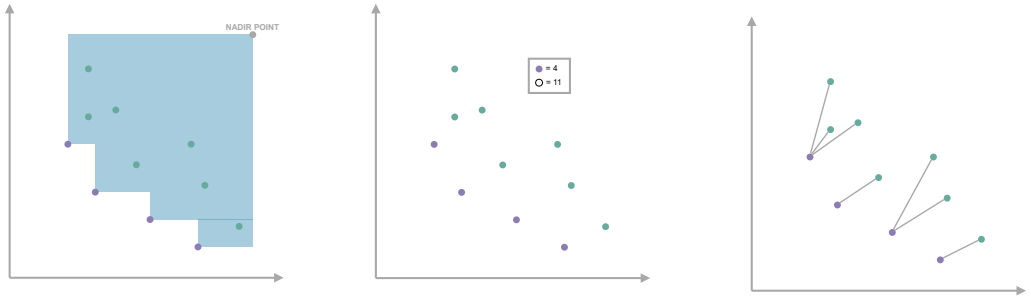
Another method allowing diverse Pareto front generation is *Conditioned One-shot Multi-Objective Search* (COSMOS) [144]. The main advantage of COSMOS is its incorporation of learning objectives trade-offs into the training process, resulting in one model that can yield all Pareto solutions. The obtainment of each Pareto front point is controlled by concatenating data with the preference vector. The penalty factor based on cosine similarity meant to broaden the Pareto front is also included in objective loss function.

2.5.3 Pareto fronts' assessment

As MOO algorithms result in several non-dominated solutions, the assessment and comparisons of the methods are not as straightforward as in the case of single objective optimisation [145]. Most of the proposed measures are based on the shape or the convergence of the estimation of the Pareto front and are calculated in relation to *perfect point*, which is the point with the best values of all criteria (that may not be possible to achieve in practice), *nadir point*, which is analogically the point with the worst objective values, and *true Pareto front* - the attainable set of non-dominated solutions, known either theoretically or calculated as the result of several algorithm runs. Depending on the knowledge of the problem and analysed properties, many different measures can be employed. One of the most popular MOO metrics in the literature [145] is *hypervolume* [146], which is the volume of the hypercube between nadir point and estimation of the Pareto front:

$$HV(PF) = \Lambda\left(\bigcup_{s\in PF}\{s'|s \prec s' \prec s_{nadir}\}\right) \tag{2.37}$$

Where PF is the assessed estimation of the Pareto front, s_{nadir} is the nadir point and Λ is a Lebesque measure, which is a generalisation of the volume. The metric increases with



(a) *Hypervolume*

(b) *Ratio of nondominated individuals*

(c) *Generational Distance*

Figure 2.16: *Visualisation of the multiobjective optimisation metrics*

the convergence of the Pareto front estimation as well as with its broadening, making it difficult to distinguish specific characteristics of the front without additional measures.

Examples of different metrics are *Ratio of non-dominated individuals* (RNI), which calculates the proportion of non-dominated solutions to all of the algorithms results, or *Generational Distance* (GD), which is the distance from the received approximation to the true Pareto front:

$$GD(PF) = \frac{\sqrt{\sum_{i=1}^{|PF|} d(s_i, TPF)^2}}{|PF|} \tag{2.38}$$

where TPF is the true Pareto front and $d(s_i, PF)$ is the distance to the closest point from the Pareto front and $|PF|$ is the size of the Pareto front approximation. While they do not carry information about the diversity of the results, they can help identify the ability to generate non-dominated solutions (in the algorithms where it is not always provided) and their convergence.

2.5.4 Multi-criteria decision making

Multi-criteria decision making (MCDM), a part of multi-criteria decision analysis, is a field dedicated to, among others, supporting the user in choosing a solution from the Pareto front. In theory, the task could be left to the users themselves, though due to the number of solutions, their similarity and trade-offs, it is usually not feasible. Decision aiding, which MCDM is derived from, recognises a few different problem types, such as the selection of a subset of appropriate actions, the classification of the alternatives into predefined groups according to their utility or ranking the solutions based on their comparisons [147]. The most basic MCDM method is the *Weighted Sum Model* (WSM) [148], which is based on assigning importances to each of the criteria. Formally, the task is to find:

$$\arg\max_{i} \sum_{j=0}^{m} a_{ij} w_j, i = 1, 2, \ldots, |PF| \tag{2.39}$$

where a_{ij} is the value of the j-th criterion of the i-th solution, while w_j is the weight assign to j-th objective. The disadvantage of the method is that it requires specific scores for each criterion's influence. Moreover, the objectives must be numeric and *comparable*, as the algorithm is based on an additive utility assumption [148].

One of the MCDM approaches is *outranking methods*, which aim to rank alternatives based on pairwise comparisons and grading of the differences in specific criterion scores. The example of such an algorithm is ELECTRE (*ELimination Et Choix Traduisant la REalite*) [149], which introduces thresholds of significant differences (both for how much aggregatively better the solution has to be in some criteria and how much in maximum can it be worse in others) as well as objective weighting. Due to this modification, even when there is technically no dominance between two solutions, there is an option to differentiate their utility. Another very popular outranking method is PROMETHEE (*Preference Ranking Organisation Method for Enrichment Evaluations*) [150]. PROMETHEE, similarly to ELECTRE, compares every pair of solutions and weighs the criteria. However, instead of defined thresholds, it assigns a preference function to every criterion, which varies depending on the difference between the value of the objective for both alternatives. Then the ranking is determined based on aggregated preferences (including the weighting) in favour of and against each of the solutions.

The MCDM field continues to grow, and many new algorithms are proposed [147]. Nevertheless, one of the requirement of the MCDM methods is that the user, called *Decision Maker*, is able to give their preferences: either in the form of specific parameters, such as criteria weights or differences thresholds [149, 150] or examples of pairs of alternatives, where one is preferred over the other [151].

2.6 Multi-objective optimisation application for imbalanced data problem

The advantage of employing multiple optimisation criteria has led to the spread of applying MOO algorithms in the imbalanced data problem. The most popular approach seems to be utilising multi-objective optimisation in creating an ensemble of classifiers. In [152], researchers analyse employing MOO in different stages of committee assembling - for generating a pool of estimators in their selection process and combination rule. E-MOSAIC [153] is the ensemble model using MOO for determining the best split of training data used to train each of the members. In contrast, [154] proposes an improvement of

the bagging algorithm by selecting appropriate bags with the utilisation of MOO. SE-MOOS [155] is an ensemble model consisting of SVM classifiers with hyperparameters and feature selection optimised by MOO. In turn, EFIS-MOEA [156] is created from estimators resulting from the Pareto front where both specific instances and features were chosen. [157] proposes a non-specialised ensemble in which members, created from different classification algorithms and bootstrapped data, are pruned by the MOO method. [158] similarly creates an ensemble from different algorithms but uses MOO for their weights assignment. The method proposed in [159] generates members by bootstrapping with undersampling while employing MOO for feature selection. Both [160] and [161] propose ensemble consisting of two classifiers - in the case [160] made from estimators trained on skewed and balanced data respectively, and in [161] of two different under- and over-sampling ensembles, and use MOO to determine their involvement. [162] proposes model using a combination of two classifiers, which are the extremes of the Pareto front, where instance selection was optimised.

A different way of employing MOO is in the preprocessing stage. An example of a data sampling algorithm is MEUS (*Multi-objective evolutionary undersampling*) [163], which selects instances from training data so that different criteria are maximised. To ensure no bias towards any of the classes, the constraint is placed so that the resulting set is balanced. In turn, [164] proposes EMDID algorithm, dedicated to selecting the best cutpoints of discretisation of continuous features.

Lastly, the other approach is to utilise multi-objective optimisation algorithms to create or train the respective estimator. The utilisation might be algorithm specific, like in the case of modifying SVM learning, so that it allows multiple losses functions [165, 166]. Another way is to employ MOO to determine the penalties of each class in the system consisting of a deep learning model [167]. The training of model parameters might be assigned entirely to the MOO algorithm, like in the case of the method proposed in [168] and [169], where it is used instead of a more typical gradient descent approach. Finally, estimator could be created based solely on the results of MOO, with the examples of [170], where optimisation algorithm is utilised for the search of different discriminant functions used for Gaussian classifiers,[171] where the algorithm chooses splits for Oblique Decision Trees, or [172], which returns the set of fuzzy rules.

The most popular choice of criteria for multi-objective optimisation algorithms for the imbalanced data problem is the base metrics connected to the quality of each of the classes' recognition. In the case of binary classification, usually *recall* and *specificity* are selected [161–163, 170, 172, 173] or the errors of each class recognition [152]. Some cases focus on minority class prediction by choosing *recall* and *precision* [155, 158, 163, 171]. Another approach is to utilise one quality metric together with diversity measure

[153, 157]. AUC is a fairly common objective [154, 156, 164, 169] paired with some system-specific indicator. Lastly, methods often include a criterion meant to decrease the computational complexity of the model, like, for example, the number of features or instances selected [152, 156, 164, 172, 174]. As for the specific optimisation algorithm, NSGA II is by far the most common choice [153, 154, 156, 162–164, 167, 168, 170, 172]. The other methods used include SPEA 2 [173], spMODE-II [152], NSGA III [171], multi-objective ant colony algorithm [159], swarm optimisation [174] and diferrential evolution [158]. Some methods evaluated all of the values from the range in case of the one parameter tuning [160, 161] or combined the criteria into one and used methods dedicated to single objective optimisation [169, 174].

Chapter 3

Application of the multi-objective optimisation in ensemble learning

This chapter aims to answer the first thesis question - Is it feasible to employ MOO in the process of training ensemble classifier, and how does it compare to the ensembles optimised using a single criterion? *For this purpose, the ensemble model will be proposed that generates the pool of estimators based on different classification algorithms and bootstrapped data.* MOO *will be employed to assign weights to each committee member, indirectly influencing their line-up. The way of comparing Pareto front-based classifiers with singular models will be also proposed. The method will be evaluated on the selection of different datasets with skewed class ratio and compared with ensembles where estimator pools will be created in the same manner, but weights will be assigned using single objective optimisation, as well as with the example of popular ensemble models to study the influence of optimised weights assignment. The results will be analysed in the context of* MOO*-based ensemble performance and its quality in reference to methods optimised with a single criterion.*

3.1 Motivation

Literature study shows that ensemble learning is a common domain of applying multi-objective optimisation. One of the approaches is to utilise an optimisation algorithm for selecting committee composition or assigning weights. Fletcher et al. [157] proposed a model in which different classification algorithms are used to build a heterogenous ensemble, further diversified by data bootstrapping. The proposed method seemed very

promising. However, despite the authors' claim that it can suit any type of data, the lack of techniques countering skewed classes' ratio and the utilisation of accuracy as one of the optimisation criteria (second being measure of diversity) discredited its appropriability for the imbalanced data problem [2], which was later backed by preliminary research. Nevertheless, some developed procedures, like estimator generation and optimisation criteria calculation, could benefit the imbalanced data ensemble model. Therefore, the method could be proposed that would incorporate the core of aforementioned model with modifications adjusting the algorithm to the context of data imbalance.

Another identified research gap that should be expanded was the analysis of the Pareto front obtained from the MOO algorithms, or more specifically, the resulted classifiers and the choice of the solution. Usually, one solution is arbitrarily selected and compared with standard SOO models. Nonetheless, one of the biggest strengths of multi-objective optimisation is the broad set of diverse solutions offered to the user. While it is difficult to compare such sets with models giving only one result, it cannot be ignored in experimental evaluation. For this reason, the way of presenting specific solutions performance, which would enable the analysis of the whole Pareto front quality, should also be proposed.

3.2 Proposed method

The main goal of the proposed method is to classify imbalanced data, though, in principle, it may also be used on balanced datasets. Three phases of training can be distinguished in the proposed approach.

- First, bagging is utilised to create a pool of base classifiers based on several different classification algorithms.
- Next, the MOO algorithm is used to produce a Pareto optimal set of classifier ensembles, jointly optimising *precision* and *recall* of the resulting model. Each ensemble constructed in this step is encoded as a vector of weights assigned to individual ensemble members.
- Lastly, since MOO methods return not one but a set of solutions, it has to be chosen which weights will be used in the final ensemble. The choice could be made by the user manually, or a predefined criterion based on the selected MCDM approach [147] may be employed, or the best ensemble could be chosen in the context of the single metric presented in (eq. 2.6-2.14).

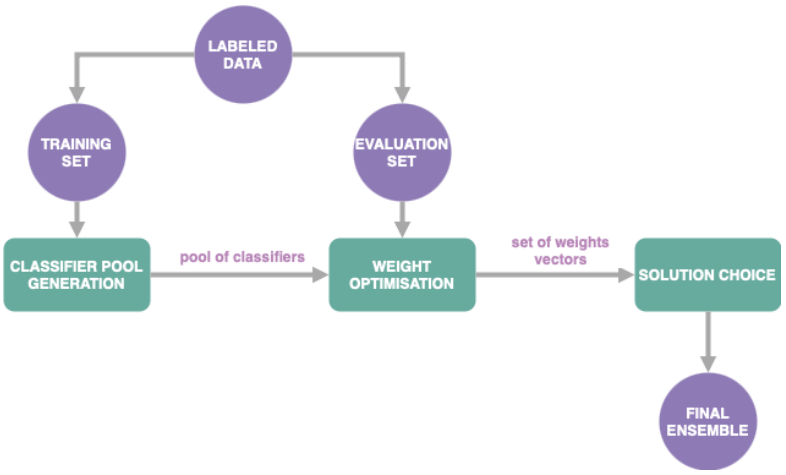


Figure 3.1: *A flow chart of the model.*

After that, the proposed model is ready to classify incoming unlabeled data, employing previously calculated weights to aggregate members' decisions during weighted voting (eq. 2.26). The flow chart of the model is presented in Figure 4.1.

Each of the phases is further described in consecutive sections.

3.2.1 Pool of classifier generation

A pool of classifiers, which become ensemble members, is generated based on the provided set of models. Models may vary based on distinctive classification methods or different parameters - in this study, the first option was chosen. Diversity is further assured by training each of the created classifiers with a distinct subset of training data - for that, stratified bagging is used. Two parameters should be provided: the number of bags, or the number of classifiers trained from a single model, and the size of the subset sampled from whole training data. The pseudocode of the process is presented in Algorithm 2.

Algorithm 1 Generating a pool of diverse classifiers

Input: $\mathcal{LS}$ - learning set
$\mathcal{TM} = \{\mathcal{TM}_1, \mathcal{TM}_2, \ldots, \mathcal{TM}_N\}$ - set of learning methods
b - number of bags per model
s - size of each bag

Output: Π - pool of trained classifiers
$\Pi \leftarrow \emptyset$
$cr \leftarrow$ class ratio of $\mathcal{LS}$
$N \leftarrow size\ of\ \mathcal{TM}$
for $i \leftarrow 1$ to N **do**
  for $j \leftarrow 1$ to b **do**
    $\mathcal{TS} \leftarrow$ set of s observations sampled with replacement from $\mathcal{LS}$ that preserve cr
    $\Psi_i^j \leftarrow$ train classifier using $\mathcal{TM}_i$ and $\mathcal{TS}$
    $\Pi \leftarrow \Pi \cup \{\Psi_i^j\}$
  end for
end for

3.2.2 Weight optimisation algorithm

The distinctive part of the proposed method is its weight optimisation algorithm. Good weight assignment is important to mute weaker ensemble members and amplify the strong ones in a process of weighted majority voting as presented in eq. 3.1.

$$\Psi(x) = \arg\max_{j \in \mathcal{M}} \sum_{i=1}^{n} p_{ji} w_i \tag{3.1}$$

Where $\mathcal{M}$ is a set of classes, w_i is the weight of the i-th classifier and p_{ij} is a support for the j-th class given by the i-th classifier. For weight optimisation, the NSGA II [134] algorithm was employed due to its popularity in the application for the imbalanced data problem as well as its lack of specific parameters, which should be provided by the experts. In the proposed method, individuals are represented by a list of real values

$$Ind = [w_1, w_2 \ldots w_N], \quad w_i \in [0, 1],$$

where w_i stands for the weight of the i-th base classifier, while N is the size of the classifier ensemble.

Two fitness functions F_1 and F_2 are proposed as follows

$$\begin{cases} \text{maximise } F_1(w_1, w_2 \dots w_N) = recall \\ \text{maximise } F_2(w_1, w_2 \dots w_N) = precision \end{cases} \tag{3.2}$$

The algorithm seeks to maximise both *precision* (2.10) and *recall* (2.8). Because these goals oppose each other, focusing on only one may lead to a situation when all or none of the samples are recognised as positive class.

3.2.3 Solution choice

The algorithm 2 returns a set of non-dominated individuals representing a combination rule of classifier ensemble (3.1). Still, finally, the single set of weights that defines the classifier ensemble must be chosen. As mentioned earlier, the solution selection from the Pareto front should be done with the cooperation of the end-user. Nevertheless, as in this study the user preference is not available, the following way of choosing solution is proposed:

- based on each of the objectives - where either *recall* or *precision* has the best value;
- balanced solution - with the smallest difference between the objectives;
- based on PROMETHEE [150] rule with usual criterion and a slight advantage of either *recall* or *precision*. The advantage of either criterion was assured by setting the weights (0.6 for the favourable objective, 0.4 for the other, so one objective is slightly more important).

3.3 Experimental study

This section presents the results of experiments that were conducted to test the quality of the designed model and compare its variants with different solution choice rules using SOO methods.

3.3.1 Objectives

The experiment aimed to answer the following research questions:

RQ1 Is it possible for the MOO-based model to outperform the quality of the models optimised in regard to its individual objectives?

RQ2 Can selected MOO-based model have a better quality than the ones created by single optimisation of aggregated metrics?

RQ3 Is there the best approach to selecting one model from the Pareto front solutions?

The consecutive segments describe utilised benchmarked datasets, the configuration of experimental studies, as well as analysis and discussion of obtained results.

3.3.2 Setup

Choice of benchmark datasets. Experiments were run on 26 different imbalanced datasets from KEEL [175], UCI [176] and Kaggle repositories. The datasets can be split into two categories: (1) small datasets (up to 1484 samples) and (2) big datasets (up to 318k samples). Chosen datasets also differ in the number of features (ranging from 3 to 187) and the Imbalance Ratio (IR), varying from 53.6% to 0.2%. Some of the datasets did not represent a binary problem. In these cases, the dataset was binarised, i.e., one class was selected as a minority class, and the rest were labelled as one majority one. The description of all datasets is presented in Table 3.1.

Implementation and reproducibility. Complete source code, sufficient to repeat the experiments, is available at[1]. The complete results of the conducted experimental analysis were also provided with the code.

The proposed algorithm, as well as the experiments described in this work, were implemented in the Python programming language. Moreover, base classifiers from *scikit-learn* module [177] were used, while the implementation of optimization algorithms was based on*pymoo* module [178].

Choice of base classifiers. All ensembles were based on the same set of different classification algorithms. The parameters of the used classifiers from *scikit-learn* module [177] were as follows:

- AdaBoost - *Decision Tree Classifier* as base estimator with *50* iterations,
- Random Forest - with *100* estimators and *Gini impurity* as split criterion,
- Naive Bayes Classifier,
- KNN Classifier with $k = 5$,
- multi-layer perceptron (MLP) with one hidden layer of 100 neurons, *rectified linear unit function* for activation and *Adam* solver for weight optimisation,

[1] `https://github.com/w4k2/moo-ensemble-weighting`

Table 3.1: *Description of datasets. #S denotes the number of samples, #F stands for the number of features, and IR indicates the imbalance ratio*

DATASET	#S	#F	IR [%]
aps failure	76000	170	1.8
covid	318438	12	2.1
credit card	284807	29	577.9
diabetes	101766	49	12.6
hand positions	78095	37	23.3
MiniBooNE	130064	50	39.0
mitbih	109446	187	2.6
page-blocks	5472	10	8.79
abalone9-18	731	8	6.1
glass4	214	9	6.5
glass5	214	9	4.4
yeast4	1484	8	3.6
yeast5	1484	8	3.1
yeast6	1484	8	2.4
flare-F	1066	11	4.2
ecoli1	336	7	29.7
ecoli2	336	7	18.3
ecoli3	336	7	11.6
glass0	214	9	48.6
glass1	214	9	55.1
haberman	306	3	36.0
pima	768	8	53.6
vehicle1	846	18	34.5
vehicle3	846	18	33.4
yeast1	1484	8	40.7
yeast3	1484	8	12.3

- Decision Tree Classifier (CART) - with *Gini impurity* as split criterion, no max depth set, miniminum samples split equal to 2 and minimum samples leaf equal to 1.

Although AdaBoost and Random Forest are examples of ensemble approaches, they are considered single estimators in this research.

The number of bags for each classifier was 3, which resulted in a pool of 18 models.

Optimisation algorithms. A standard genetic algorithm (GA) was used for single objective optimization, with a tournament selection, two-point crossover and a Gaussian mutation. For both NSGA II and GA, the number of iterations was set to 500, the population consisted of 500 individuals, and the probability of mutation was 50%.

The following objectives for SOO-based ensembles were chosen: *precision*, *recall*, BAC, *F1 score*, *Gmean* and AUC.

Solution choice rules.

As mentioned before, presented models were chosen based on the best objectives values (labelled MOO *precision* and MOO *recall* respectively), smallest difference between criteria scores (MOO *balanced*) and two according to PROMETHEE ranking - one with *recall* - *precision* weights being equal to 0.6 and 0.4 (MOO PROMETHEE *recall*) and second with reverse weights (MOO PROMETHEE *precision*).

Data partitioning. Inside the model, data was divided into training and validation sets (used during the composition optimization process) with a ratio of 70:30%. The experiments were conducted using 5×2 cross-validation, and the presented results are the average value of all the metrics from individual folds.

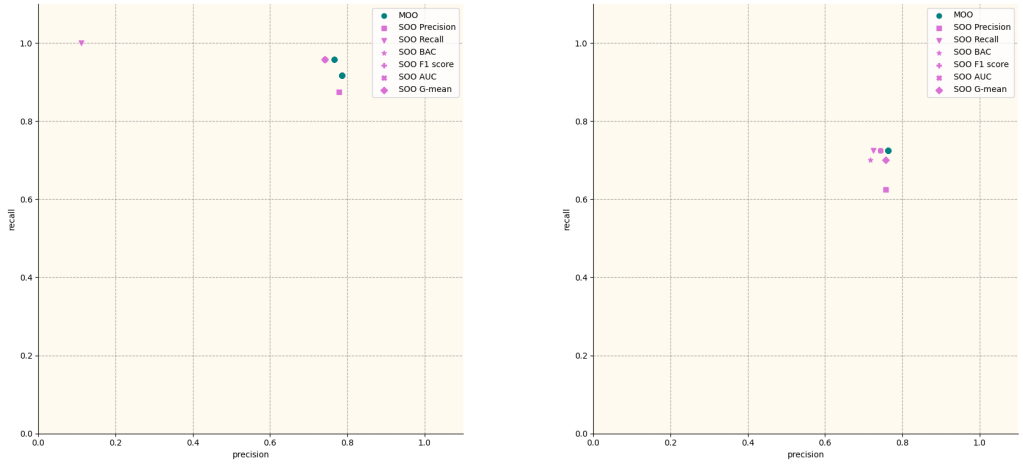
Model comparisons Experiments were divided into two parts. In the first part, the results of the MOO and SOO optimisation algorithms were compared and presented in the form of Pareto fronts. For the second segment of overall classification performance analysis, two comparative methods of the AdaBoost model built on decision trees and the Bagging Ensemble consisting of 18 members (the same number as proposed method) were included. Both methods are popular examples of widely utilised ensemble models without a particular usage of data preprocessing algorithms, as they were not an object of this research and could disrupt the comparison.

Result analysis. To analyse the classification performance of the chosen algorithms, the Friedman test and Nemenyi post hoc test at a significance level of 0.05 were chosen [179].

3.3.3 Pareto fronts' analysis

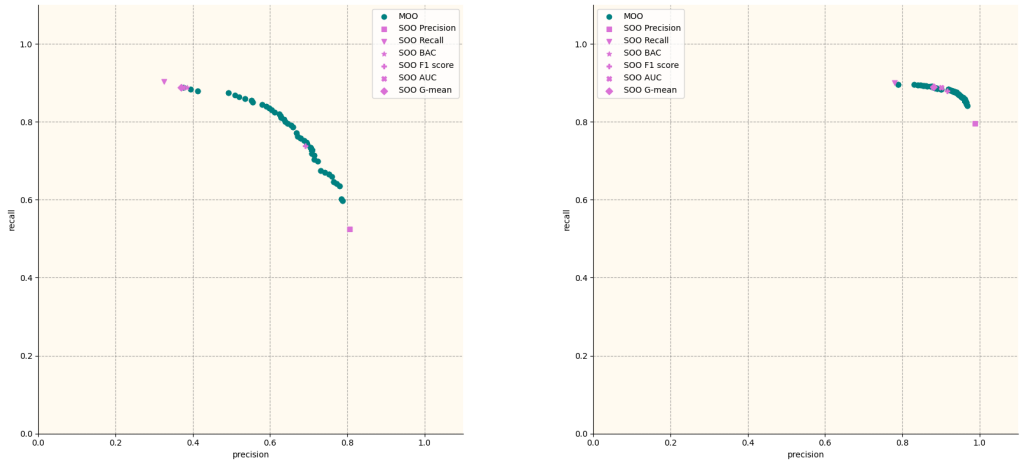
Figures 3.2 and 3.3 present examples of Pareto fronts obtained from the MOO algorithm together with scores of each objective for solutions of single-objective optimization.

It may be observed that for big datasets, the Pareto front is well-defined and has a lot of diverse solutions. Most SOO solutions lie directly on or at a small distance from the front. The only exceptions are solutions optimised concerning both objectives, that are *precision* and *recall*, which sometimes lay far away from the rest. However, such models would not be acceptable in realistic scenarios because they are biased in predicting too many or too few samples as a minority class (as the other objective is very low). In the case of smaller datasets, the Pareto front is typically very constrained (in the worst

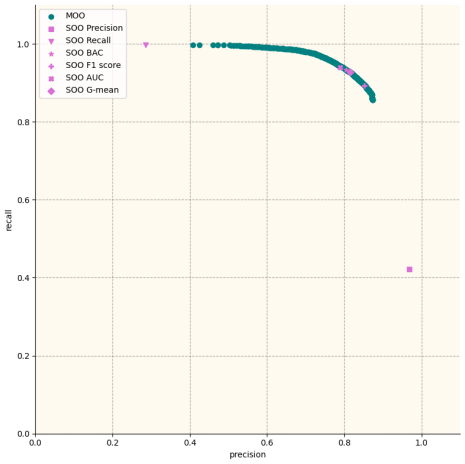


(a) *yeast3* **(b)** *pima*

Figure 3.2: *Examples of Pareto front of small datasets.*



(a) *aps failure* **(b)** *hand positions*



(c) *MiniBooNE*

Figure 3.3: *Examples of Pareto front of big datasets.*

case, the algorithm returned either only one solution or a few solutions but with the same objective values). This phenomenon is probably due to limited possible values of objective functions, which rely heavily on the minority class's size, further limited by employed stratified split into training and validation sets. This also means that criteria are less stable and prone to overfitting. Moreover, the rest of the single-objective solutions are distributed irregularly, though, with few exceptions, in a similar area. Even though the Pareto front is relatively tight, the solutions obtained are considerably balanced in terms of objectives and, in some cases, are even better than the ones given by SOO. Still, this suggests that the method requires a reasonably large dataset to achieve a satisfactory performance, particularly a well-defined, wide front. Finally, it is worth noting that the single-objective solutions typically have a different position on the Pareto front, indicating that the optimal *precision*/*recall* trade-off depends on the specific performance metric choice. This, in turn, suggests the usefulness of multi-objective optimization, which can simultaneously produce multiple solutions optimised to the particular metrics.

3.3.4 Classification performance analysis

The detailed results of all quality measure scores are presented in Tables 8.1 - 8.6, while Figures3.4 and 3.5) show the average ranks of methods and statistical evaluation. Firstly, it should be analysed how MOO-based solutions compare considering aggregated metrics with ensembles optimised especially for these quality measures. In general, models generated from the Pareto fronts and selected according to best *recall* (MOO *recall* and PROMETHEE *recall*) as well as the balanced solution (MOO *balanced*) outperform the single optimisation estimators. *Recall*-based MOO solutions expectedly achieve the highest average rank in the case of BAC and AUC, while MOO balanced comes in terms of *F1 score*, where *precision* has a bigger influence than the rest of the aggregated metrics. The SOO aggregated metrics ensembles do achieve the highest score for their respective quality measures in some datasets, though in general they rank lower and are not consistent with their optimisation criteria. This might be due to weaker generalisation abilities and some overfitting. However, it should be noted that the difference is not statistically significant (Fig. 3.5).

The situation is different when analysing the results of simple measures. Here, the ensembles based on single objective optimisation according to *precision* and *recall* obtained the highest scores of their respective metrics by far, both in average ranks and number of datasets for which they achieved the best results. MOO *precision* model performed similarly to its SOO counterpart, since its scores were often very close or even better for some problems. MOO *recall*, even though second in average rank to SOO *recall*, has never exceeded it. Nevertheless, both MOO *recall* and *precision* (as well as solutions selected

with PROMETHEE) obtained much more balanced results compared to SOO ensembles. For single objective approaches, their excellent results in their respective metrics go together with very low, usually the worst, scores of the opposite measure. This indicates that the methods are strongly biased towards or against the majority class and should not be employed in the pattern recognition task. On the other hand, MOO solutions, even when selected according to the best *precision* and *recall* values, still achieved more balanced results, with less of a difference between these two metrics. It is also reflected in their aggregated measures scores.

When comparing the proposed ensemble with established methods of AdaBoost and Bagging, it should be noted that the latter seems to be generally inferior for almost all selected metrics. Only for *precision* and *F1 score*, which is highly dependent on the preceding, does an ensemble using standard bagging obtain results better than most of the MOO-based models. Nevertheless, it must be highlighted that for all the metrics, a variant of the proposed method with a higher average rank can always be chosen, though the difference is statistically significant only in the case of recall values. Regarding AdaBoost, both figure 3.4 and statistical test results show that most of the MOO-based methods are of better quality, where for *precision*, *recall* and *F1 score*, the difference is statistically significant.

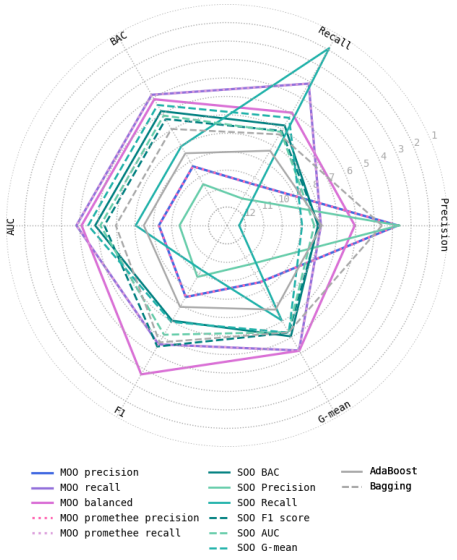
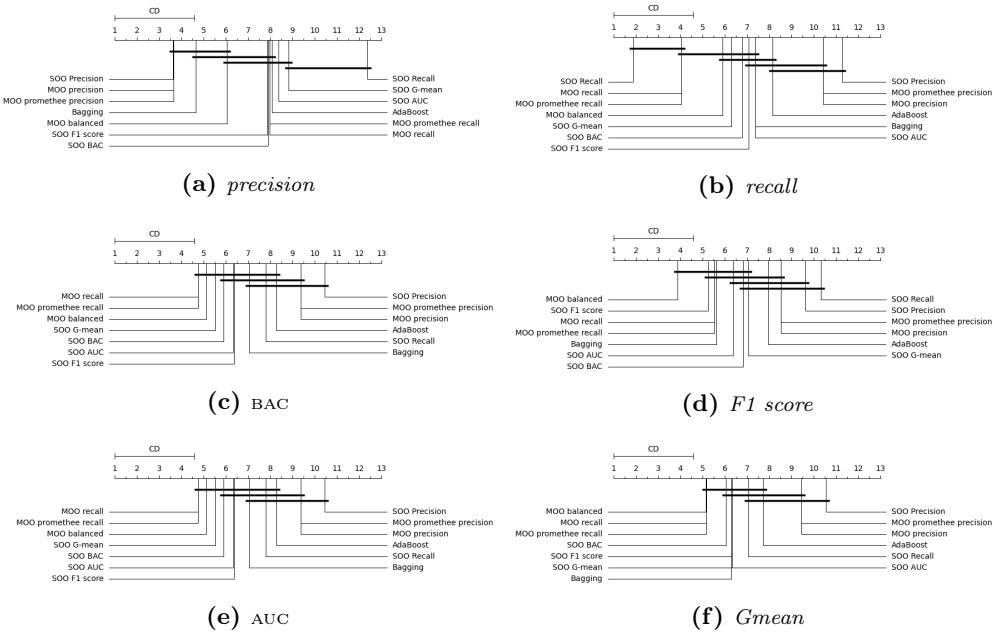


Figure 3.4: *Average rank of every optimisation method for different performance metrics.*



(a) *precision* **(b)** *recall*

(c) BAC **(d)** *F1 score*

(e) AUC **(f)** *Gmean*

Figure 3.5: *Results of Friedman and post hoc Nemenyi statistical tests for different metrics*

3.4 Lessons learnt

In this chapter, the ensemble model utilising MOO for weights assignment was proposed. The pool of classifiers was generated utilising several classification algorithms, as well as stratified data bootstrapping. Optimisation objectives were calculated based on the validation set separated from the training. Analysis of the resulting Pareto fronts demonstrated that for small datasets with low minority class count, solutions tend to converge to a single point due to limited ranges of possible criteria values. This phenomenon may lead to overfitting and significantly hinders the choice of solutions from the set.

As for the solution selection method, the results of the experiments show that there is no one best way of choosing individual weight vectors from the Pareto front. Depending on the metrics, either of the proposed solutions proved the best. However, the differences that were obtained were not always statistically significant. It is worth noting that solutions selected via the PROMETHEE method were the same as the ones with the highest respective objective. It is probably caused by a small number of criteria and too general assumptions of the algorithm rules. For this reason, the method was not chosen in further research.

The proposed committee was compared with ensembles, in which weights were optimised according to one criterion. One of the MOO-based models was generally better for every metric, with the only exception being *recall* and *precision*. Generally, ensembles optimised utilising MOO were characterised with more balanced results, even the ones with

weights chosen based on the best objective scores. This could be especially noticed with the values of aggregated measures, where both factors (i.e. some base metrics) had the same influence on the results. However, it must be pointed out that even though the general rank of MOO ensembles was higher, they did not always obtain the best results for every tested dataset.

Lastly, the proposed model was compared with different ensemble approaches, namely Bagging and AdaBoost. In general, the MOO-based committee obtained better ranks. However, there were datasets where one of the compared ensembles was the best. Still, the comparison is not exactly straightforward. Even though the compared method did not have any specific techniques to counter data imbalance, such as member weights optimisation, they still use procedures which might help with better minority class recognition. For this reason, it cannot be fully determined which factor had the biggest influence on the final results, and better quality might be problem-specific.

The proposed method, together with the results of the experimental evaluation, was published in [180].

Chapter 4

Application of multi-objective optimisation in data sampling

This chapter focuses on the application of MOO *in data preprocessing, as to address the second research question posed in the thesis* - Is it possible to employ MOO in the preprocessing stage, and how does it improve the quality of the estimator?. *For this purpose, a hybrid oversampling algorithm will be proposed that creates several neighbourhoods of minority class samples, conducts cleaning, and generates new observations. The proposed method will be assessed utilising a* CART *classifier and compared to the estimator trained on original data, as well as data sampled by different popular algorithms.*

4.1 Motivation

Data preprocessing is the most popular approach to countering the problem posed by data imbalance. Its main advantage over other methods is its universality, independence from specific classification algorithms, as well as lack of interference in predictive models or reliance on expert knowledge [181]. Nevertheless, as the previous research suggests, MOO is not widely used in the design of such algorithms. The only usages of multi-criteria optimisation for data preprocessing found in the literature were for feature [155, 156, 159] or instance [156, 162, 163] selection, which allows solely the undersampling that is not always feasible. Previous research showed that employing MOO in an imbalanced data classification model gives an advantage over utilising aggregated metrics and results in more balanced solutions in relation to minority class prediction quality and bias. Thus, it could be beneficial to apply it for the new sample generation to populate the areas that demand better minority class representation and not to create detrimental noise.

CCR (*Combined Clearning and Resampling*) [98] is a hybrid sampling method that removes the majority class samples and generates synthetic minority class samples in close proximity to every minority class observation. The number of generated samples, as well as the area where the procedure will be conducted, is determined by the neighbourhood of each sample so that classes are better separated and no noise is enhanced. This approach could be improved by optimising the radii and populations of the areas based on the actual classification quality, as MOO employment should also ensure amplification of the minority class influence without overly impairing minority class recognition. However, it is important to modify the way the neighbourhoods are determined, since cleaning and resampling areas around single points may lead to overfitting and ignoring parts of feature space with smaller minority class representation, which could be lost due to data partition. For the same reason, the evaluation protocol of the optimisation algorithm should also be adjusted.

4.2 Proposed method

The proposed method is based on the CCR idea of cleaning neighbourhoods of the minority class samples from the majority class and generating new objects. However, the areas are determined differently, so they depend more on the density of the majority class rather than specific observations. Furthermore, the size of each neighbourhood and the number of generated samples are optimised employing a MOO algorithm to fit the problem and eliminate areas that could deteriorate classification quality.

The idea of the algorithm is presented in Fig. 4.1.

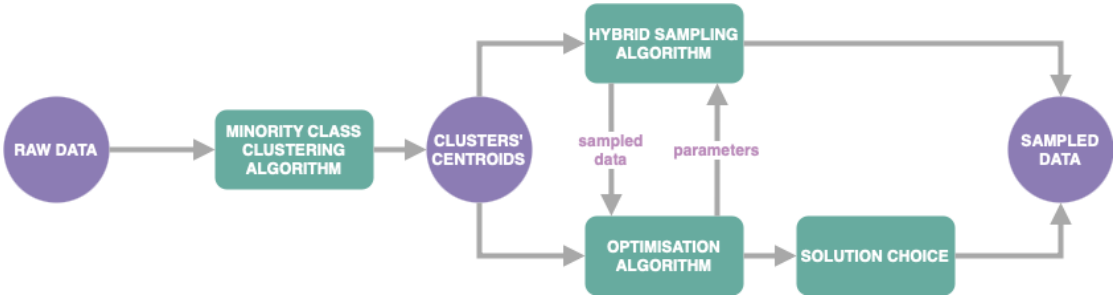


Figure 4.1: *A scheme of the proposed method*

Each of the subsequent part of the proposed algorithm are further described in the following sections.

4.2.1 Neighbourhood determination

The first step of the algorithm is to determine the neighbourhoods, where cleaning and generating samples will be conducted. The algorithm employs areas around groups of minority samples, which are determined by the clustering method - *K-means* algorithm to ensure possibly high generalisation and avoid overfitting. Calculated centroids of the clusters become centres of neighbourhoods, defined as n-dimensional spheres, where n is the problem's dimensionality. Radii of determined areas are optimised at a later stage of the algorithm.

K-means algorithm requires defining the k parameter, though any appropriate number of clusters may vary based on the dataset size or distribution of the objects [77]. In the conducted experiments, $k = \lceil \frac{1}{3} N_{min} \rceil$ centres were chosen empirically, where N_{min} corresponds to the number of minority class samples used for training. This number was found to be a good trade-off, so the selected clusters do not contain too few minority class samples and give the optimisation algorithm the flexibility in adjusting the neighbourhoods and detecting noise samples.

4.2.2 Hybrid sampling algorithm

The main part of the proposed method is a process of cleaning and generating data. The algorithm removes every majority class object for each given neighbourhood. Then, depending on how many majority class samples are removed, an appropriate number of minority class samples within the spheres are randomly generated. The number of samples for each area is calculated based on the parameter $frac$, indicating what percentage of overall samples should be generated inside the given area.

The procedure is described in the Algorithm 2.

4.2.3 Optimisation algorithm

The last part of the proposed method is the optimisation algorithm. The optimisation step aims to differentiate between rare samples, which should be enhanced, and noise that should be avoided, as well as broaden the border between classes by selecting sizes and sample counts of neighbourhoods that would be the most beneficial for the predictive abilities of the classifier. To select parameters that are best tailored for the classifier, it was decided to incorporate them in the optimisation process to properly estimate the quality of the prediction from training on the resampled data.

Algorithm 2 Hybrid sampling algorithm

Input: $centers$ - set of coordinates of sphere centers
r - set of spheres radii
f - set of percentages of samples to be generated
X_{maj} - set of majority class samples
X_{min} - set of minority class samples
Output: $\mathcal{LS}_{new}$ - resampled data set

```
n ← size(centers)
for i ← 1 to n do
    for x ∈ X_maj do
        if d(x, c_i) ≤ r_i then
            X_maj ← X_maj \ {x}
    end for
end for
n_gen ← size(X_maj) − size(X_min)
for i ← 1 to n_gen do
    n_i ← n_gen * f_i
    for j ← 1 to n_i do
        x_new ← randomly drawn sample from the ith sphere
        X_min ← X_min ∪ {x_new}
    end for
end for
LS_new ← X_maj ∪ X_min
```

The optimised parameters are in the form of a real value vector

$$Ind = [r_1, r_2, \ldots, r_k, f_1, f_2, \ldots, f_k]^T$$

where r_i represents the radius of the ith sphere and f_i stands for the percentage of minority class samples, which will be generated in the sphere (parameter *frac*). The two proposed objectives of the optimisation to be maximised are previously employed *recall* and *precision*. With the aim to prevent the overfitting of the model, the values of the criteria were obtained employing a 5x2 cross-validation protocol.

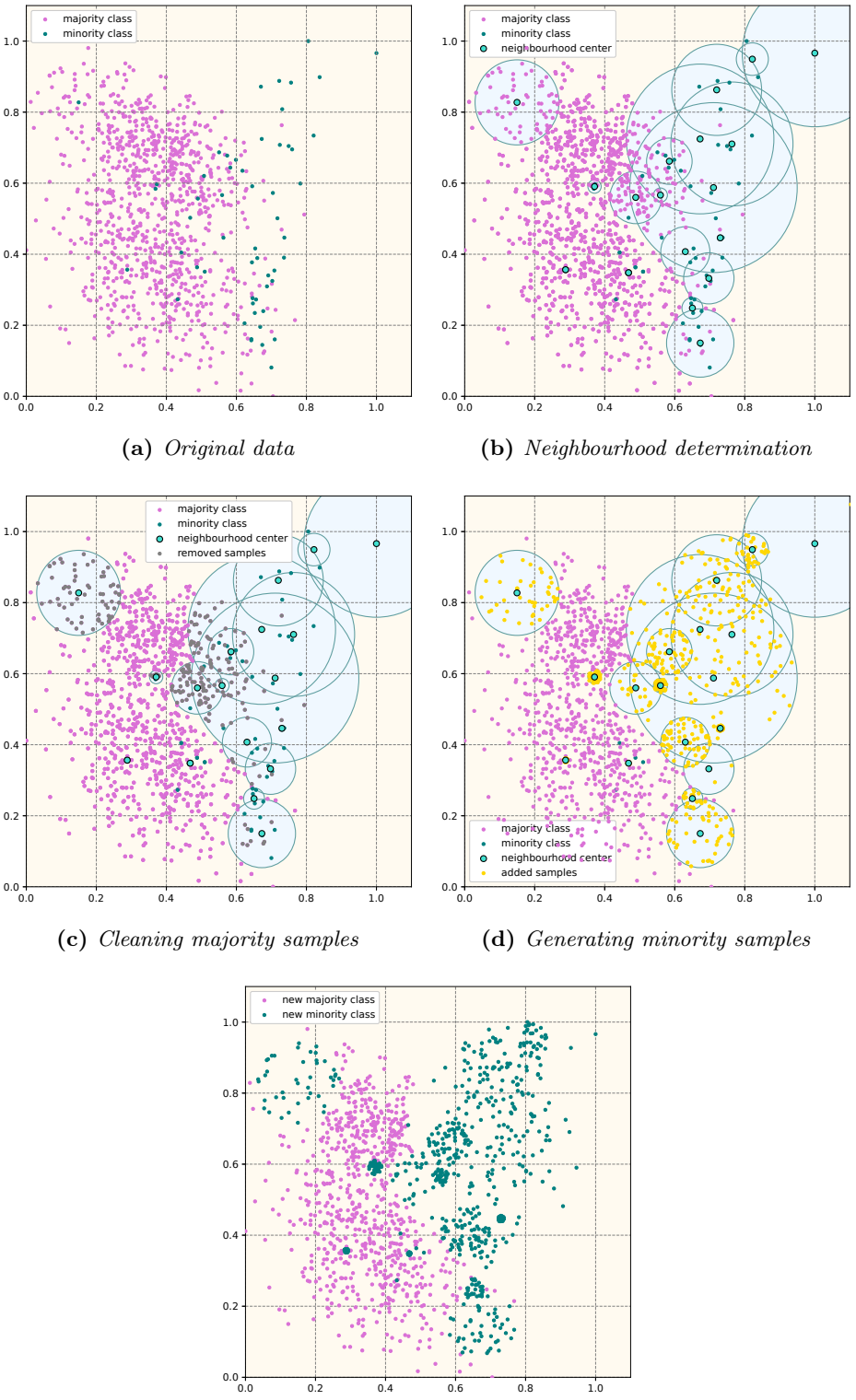
The example results of consecutive algorithm steps are depicted in Fig. 4.2.

4.3 Experimental study

4.3.1 Objectives

The experiments were designed to answer the following research questions:

RQ1 What are the properties and the quality of the proposed method?



(a) *Original data*

(b) *Neighbourhood determination*

(c) *Cleaning majority samples*

(d) *Generating minority samples*

(e) *Resampled data*

Figure 4.2: *Example of algorithm processing*

RQ2 How does the proposed algorithm compare to the state-of-the-art sampling methods?

4.3.2 Setup

Choice of benchmark datasets. The experiments are conducted using 36 benchmark datasets from the KEEL repository [175]. All of the datasets represent binary problems. The datasets vary in size (both in samples and dimensionality) along with different imbalance ratios (ranging from 2.06 to 85.88). Considering experimental protocol and inner data division caused by fitness function, it was decided to employ datasets where the minority class size is at least 17 samples. Descriptions of the used datasets may be found in the Table 4.1.

Table 4.1: *Description of datasets.* #S *denotes the number of samples,* #F *stands for the number of features, and* IR *indicates the imbalance ratio*

DATASET	#S	#F	IR [%]	dataset	#S	#F	IR [%]
page-blocks-1-3 vs 4	472	10	15.86	*yeast-0-5-6-7-9 vs 4*	528	8	9.35
yeast-1-2-8-9 vs 7	947	8	30.57	*yeast-1-4-5-8 vs 7*	693	8	22.1
yeast-1 vs 7	459	7	14.3	*yeast-2 vs 4*	514	8	9.08
yeast-2 vs 8	482	8	23.1	*yeast4*	1484	8	28.1
yeast5	1484	8	32.73	*yeast6*	1484	8	41.4
ecoli-0-1-4-7 vs 2-3-5-6	336	7	10.59	*ecoli-0-1 vs 2-3-5*	244	7	9.17
ecoli-0-2-6-7 vs 3-5	224	7	9.18	*ecoli-0-6-7 vs 3-5*	222	7	9.09
ecoli-0-6-7 vs 5	220	6	10.0	*yeast-0-2-5-6 vs 3-7-8-9*	1004	8	9.14
yeast-0-3-5-9 vs 7-8	506	8	9.12	*abalone-17 vs 7-8-9-10*	2338	8	39.31
abalone-19 vs 10-11-12-13	1622	8	49.69	*abalone-20 vs 8-9-10*	1916	8	72.69
flare-F	1066	11	23.79	*kr-vs-k-zero vs eight*	1460	6	53.07
poker-8-9 vs 5	2075	10	82.0	*poker-8-9 vs 6*	1485	10	58.4
poker-8 vs 6	1477	10	85.88	*winequality-red-4*	1599	11	29.17
winequality-white-3-9 vs 5	1482	11	58.28	*winequality-white-3 vs 7*	900	11	44.0
ecoli1	336	7	3.36	*ecoli2*	336	7	5.46
ecoli3	336	7	8.6	*glass0*	214	9	2.06
glass1	214	9	1.82	*haberman*	306	3	2.78
pima	768	8	1.87	*yeast3*	1484	8	8.1

Parameter setting. The only parameters of the proposed method arise from the application of the evolutionary optimisation algorithm. For the NSGA II [134], the population was set to 400 individuals, the number of iterations was 1000, and uniform crossover and Gaussian mutation were used.

Solution choice rules. Just as in the previous research, for the sake of comparisons three solutions were chosen from the estimation of the Pareto front - two with the highest value of each criterion (*best precision* and *best recall*), as well as the solution, where the difference between criteria was the smallest (*balanced*).

Benchmark algorithms. The proposed method was compared to the selected baseline sampling algorithms with the following parameters:

- RandomOversampler (ROS),
- SMOTE [81] with $n = 5$ neighbours,
- Borderline SMOTE [83] with $n\_neighbors = 5$ and $k\_neighbors = 5$,
- ADASYN [84], with $d\_th = 0.9$,
- CCR with $energy = 0.25$ and $scaling = 0.0$.

Each of the algorithms samples objects until the point of equal classes' sizes.

Tested classifier The proposed method and compared algorithms were employed to sample data and then used to train a CART classifier, with *Gini impurity* as a split criterion, no max depth set, and assessed via its predictive abilities.

Implementation and reproducibility The source code of the proposed method and conducted experiments are available at the online repository [1]. All of the methods and procedures were implemented in the Python programming language, employing *pymoo*[178], *imbalance-learn* [182], *smote-variants* [183] and *scikit-learn* [177] modules.

4.3.3 Results

4.3.3.1 Proposed method performance

To answer **RQ1**, the analysis of the method's obtained result was conducted. The detailed results are presented in the Appendix (Tables 8.7-8.11). It must be noticed that the algorithm's preprocessing improves the level of minority class recognition (measured by *recall*) for almost all of the evaluated datasets compared to the original data. Moreover, in the case of other measures (excluding *precision*), there is almost always at least one, but very often even all of them, solution that exceeds a classifier trained on original dataset. The only exception is *precision*, which is to be expected, since the increase in minority class prediction goes in pair with the precision deterioration, and to some extent, *F1 score*, where *precision* is equally influential as recall. Nevertheless, all of the selected solutions obtain better results on average according to *recall*, BAC, *Gmean*, and *F1 score*, which is substantiated with statistical evaluation for the first three metrics (Fig. 4.4).

[1] `https://github.com/w4k2/moo-sampling`

As for the quality of specific solutions, just as before, the solutions picked based on one of the optimisation criteria tend to perform the best according to the same measures. However, there are examples of the best precision solution outperforming the best recall solution regarding the latter measure, but never the other way around. Surprisingly, balanced solution seems to perform generally the worst across all of the selected metrics, excluding *precision*. Nevertheless, there are still datasets in which it gets the best performance. All in all, the solution selected based on the highest *recall* seems to perform the best (meaning obtaining the highest quality on the most datasets, as well as the highest average rank) according to *recall*, BAC, *Gmean*, and *F1 score*, though the difference is not statistically significant. These results might indicate that the estimations of the quality of the solutions obtained during optimisation are not very precise. This might happen due to the data partitioning during objectives calculation and resulting minority class granulation, as well as the nature of data sampling, which might lead to overfitting and thus to worse estimation of the model's actual quality.

4.3.3.2 Algorithms comparison

Finally, to answer **RQ2**, it is important to determine how the proposed algorithm compares with other oversampling methods. It can be noticed that for all of the metrics, excluding *precision*, one of the method solutions obtained the best results for most of the tested datasets, while sometimes even all three of the proposed variants exceeded or were close to the baseline. The biggest improvement of the predictions quality took place in the case of the datasets *poker-8-9 vs 6* and *poker-8 vs 6*, where proposed algorithm obtained results better by even 20 percent points, however, there were also instances where the increase of quality measure was of the order of a few percent. In general, the proposed method obtains the best average ranks in the case of *recall*, *Gmean*, BAC, and *F1 score*. All selected solutions present significantly better performance, considering at least one assessment metric, than SMOTE, Borderline SMOTE, ADASYN and ROS. Compared to the most similar algorithm, CCR, the proposed method outperforms it in the case of many datasets and has a generally higher rank. The only exception is *precision* measure, however, there are still some datasets where optimised sampling proved superior. Nevertheless, the differences are not statistically significant, so the employment of either method should be considered based on the specific problem and specifications.

It is also worth noting that the trade-off between *recall* and *precision* of the proposed approach is the smallest among all the assessed techniques. It is especially visible in comparison to SMOTE and ADASYN, which have high values of *recall* and the rest of similar quality measures while also obtaining the worst scores of *precision* and *F1 score*.

This indicates that while the proposed method has good generalisation abilities, it does not occur at the cost of decreased recognition of majority class objects.

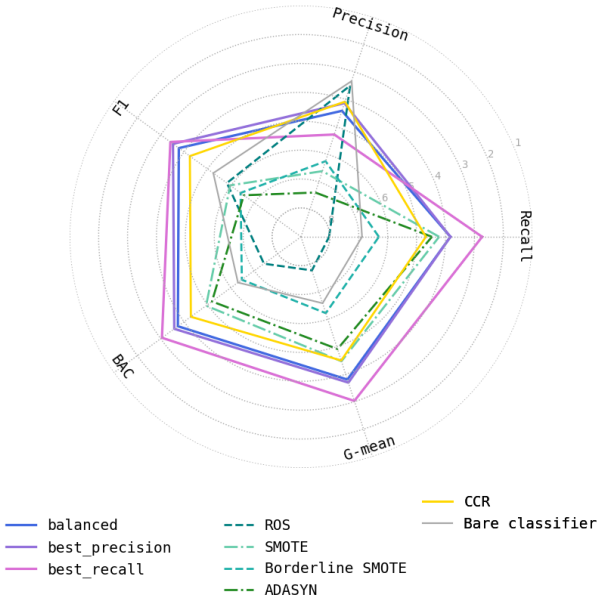
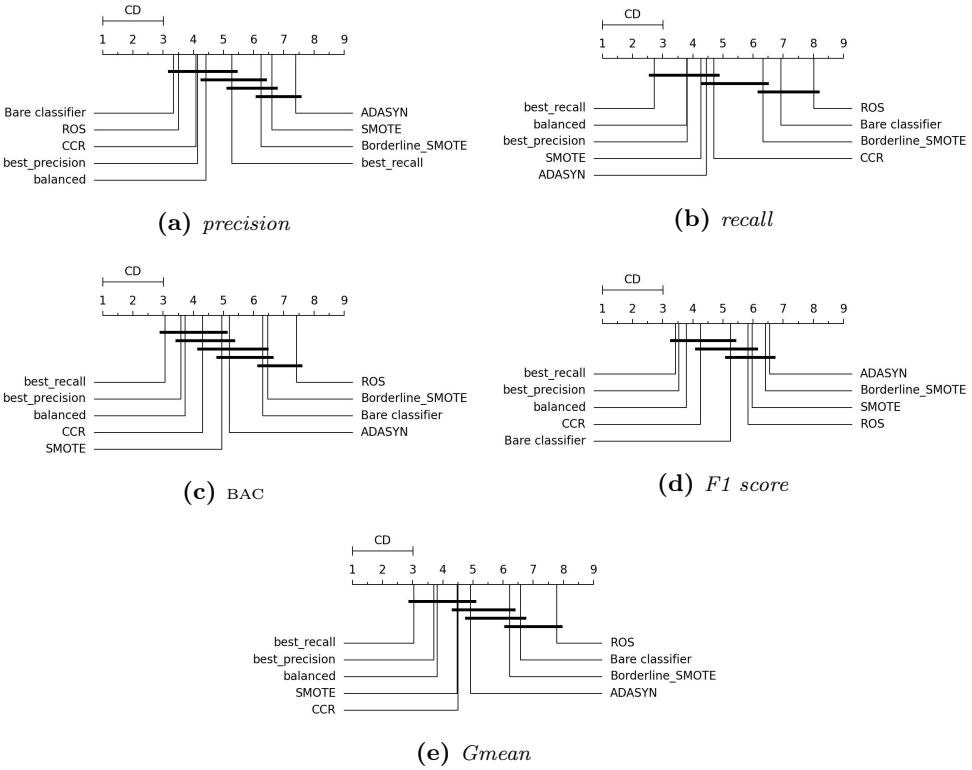


Figure 4.3: *Average rank of every sampling method for different performance metrics.*



(a) *precision* **(b)** *recall*

(c) BAC **(d)** *F1 score*

(e) *Gmean*

Figure 4.4: *Results of Friedman and post hoc Nemenyi statistical tests for different metrics*

4.4 Lessons learnt

This chapter proposed the sampling method utilising multi-objective optimisation. The algorithm removes the majority class from the determined minority class neighbourhoods and populates them with new synthetic samples. The sizes of these areas, as well as the number of generated samples, are optimised according to the best values of *precision* and *recall* metrics. As creating new samples based on hold-out assessment might lead to excessive overfitting, the objective values are calculated employing the 5x2 cross-validation protocol.

The experiments that were conducted showcased frequent improvements in the quality of the classifier trained on the data sampled by the proposed method compared to other popular sampling algorithms. The obtained results were characterised by more balanced *recall* and *precision* values, indicating good generalisation without intolerable focus on minority class samples. Obtained MOO solutions tended to act similarly to their Pareto front equivalents in the case of the edge instances. However, balanced solutions did not always have this property. In general, the datasets sampled according to the highest recall obtained the best results across most of the metrics, though there were instances where other solutions performed better. This indicates that selecting the parameters from the estimation of Pareto front should be backed by some further analysis or assessment. Nevertheless, all of the solutions obtained very good results and, in many cases, outperformed the baseline approaches.

It must be highlighted that the obtained results are indicative only of the good quality of the proposed method when used with the CART classifier, since only such evaluation is presented in this work. Nonetheless, based on the growing demand for understanding the model decision, interpretable methods such as decision trees become more desirable. This feature can also facilitate analysing and selecting the solutions from the resulting estimation of Pareto front, since their objective values might not always indicate the future performance. Moreover, computational complexity of the proposed methodology is very high, since data preprocessing and classifier training must be repeated several times to obtain the fitness function values, further amplified based on the chosen optimisation population size and number of iterations. Because of that, employing more computationally complex classifiers, such as SVM, might not be feasible.

The presented method and experimental evaluation were published in [184].

Chapter 5

Analysis of the fitness calculation protocols

This chapter aims to answer the third research question - What is the best approach to estimate the quality criteria of the classifiers built using MOO? *For this reason, the experiment was conducted comparing three methods of estimating optimisation objectives: holdout, testing on training set and 5x2 cross-validation protocols. These approaches were employed to assess the quality of the classifier trained with undersampled data, where the specific training instances were selected via the optimisation algorithm. The methodology was tested using examples of different classification models. The analysis was conducted to determine the performance of the tested approaches, considering the quality of estimated Pareto fronts, the estimation of the assessment on the test data and actual predictive abilities.*

5.1 Motivation

Employment of an appropriate evaluation protocol is crucial to properly estimate the quality of the model and thus obtain credible results. A discussion about various test sampling approaches has long been present in the literature, and their strengths as well as limitations are generally understood and sometimes, unfortunately, exploited. Nevertheless, this subject is not covered at all in the context of the optimisation objective function calculation, even though metrics employing classifiers' predictions are very often selected as optimisation criteria. In the standard optimisation tasks, objectives appear to be deterministic. However, this is not the case with measures such as recall or precision, where it is only possible to obtain an estimation of their values. In the MOO application

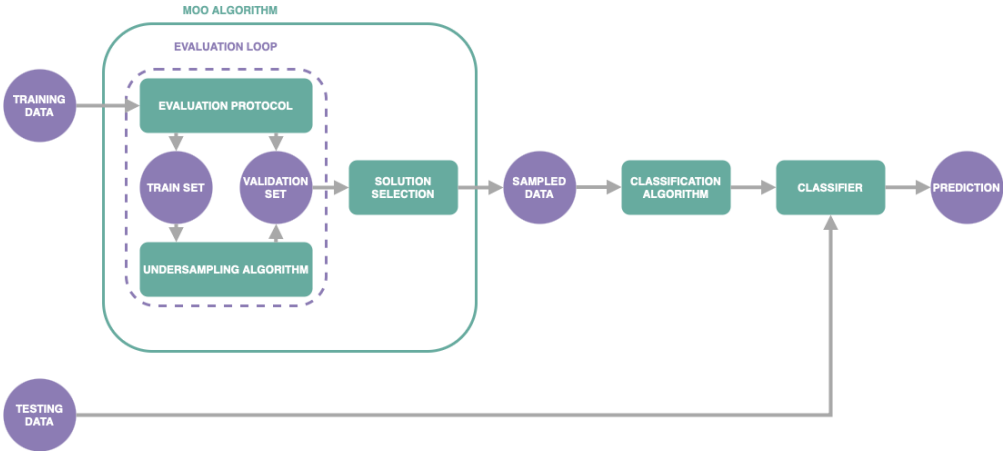


Figure 5.1: *The idea of the studies*

for imbalance data validation, using a hold-out protocol seems to be the most popular [157, 162, 163, 169, 185], but there are also works that calculate quality measures based on the whole training set [170, 173]. Alas, very often, the information about the way criteria are obtained is not disclosed.

Previous research reflected the problem with appropriate objectives estimation, since the difference between assessments conducted on separate test sets and those from the optimisation algorithm was sometimes very prominent, i.e., the solution with the highest score of one criterion did not obtain the same results when tested outside the optimisation loop. With the imbalance in the data, another challenge arises when already under-represented classes are split into smaller segments, which could lead to poor generalisation of the sample distributions. Moreover, some utilised protocols might result in overly optimistic objective scores, especially testing on training set, since high metric values might be achieved due to model remembering objects, not learning from them. For this reason, it should be studied how different objective calculation methods influence the set of solutions and which protocol is the most fitting for the estimation of the model created employing MOO methodology.

5.2 Methodology

This study aimed to compare different ways of obtaining optimisation objectives. Each of the studied protocols was used to estimate the quality of the classifier trained using data that was the result of the optimisation. The general process of the research is presented in Fig. 5.1, while each of the parts is described in more detail in the consecutive sections.

5.2.1 Optimisation algorithm

To study the influence of the objective estimation selection, a simplified version of the undersampling method found in the literature [156, 163] was employed. The goal of the optimisation is to select samples from the training data that will be used to train the chosen classifier. The algorithm individuals have a form of a vector

$$[u_1, u_2, \ldots, u_k] \quad u_i \in \{0, 1\}$$

where $u_i = 0$ means that the ith sample was removed from the set, and $u_i = 1$ denote its usage in training of the classifier. There is no mechanism guaranteeing the class's balance, meaning that potentially all of the samples of one of the classes could be removed.

Just like the previous research, the two optimised criteria are *precision* and *recall*.

5.2.2 Tested protocols

In the study, three different evaluation protocols 2.2.2 were employed to estimate the objectives:

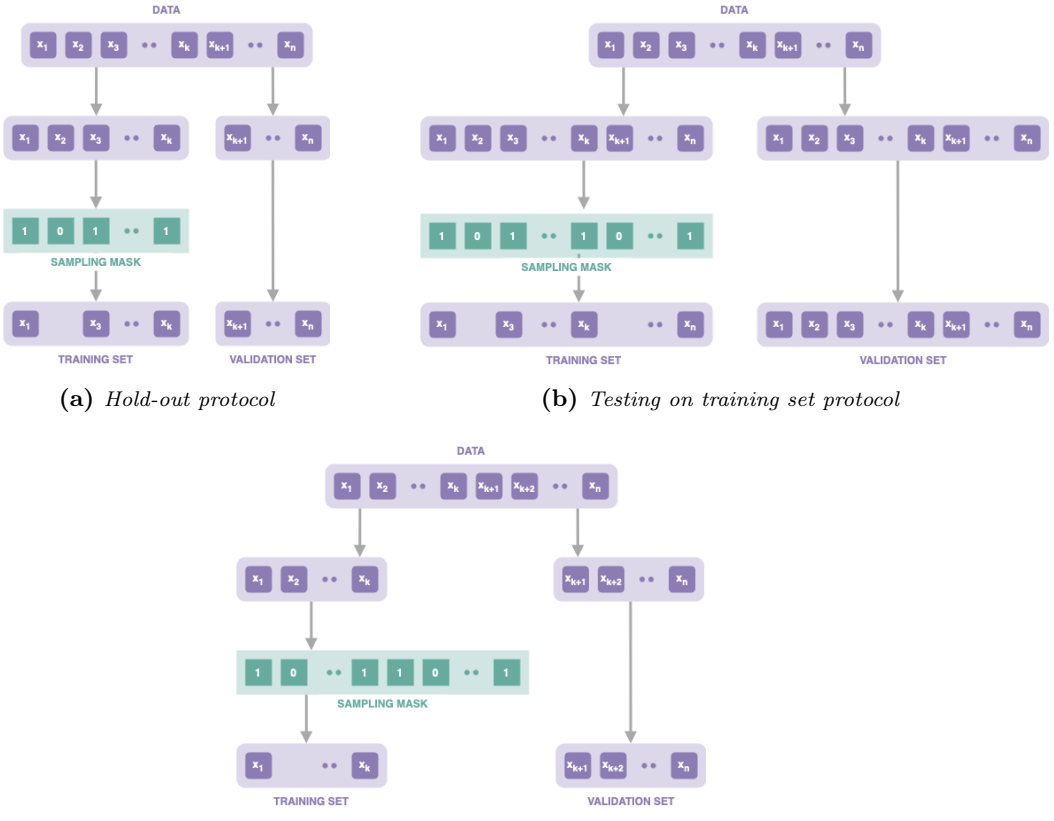
1. Hold-out protocol [20]- which seems to be the most popular in the application of MOO for imbalanced data;
2. Testing on training set [8] - which cannot usually be employed in the model assessment, however, is sometimes utilised during optimisation;
3. 5x2 cross-validation [23]- which is widely popular in classifier testing.

The individuals of the optimisation algorithm, as well as their processing, are customised to match the specifics of the data divisions:

1. For the hold-out protocol, the validation set is separated before the optimisation and passed as an argument directly for the objectives calculation, while the individuals have the size of the remaining samples. The final solution is applied only for the designated training set, and the validation set is not used for later classifier training.
2. In the case of testing on training set, the mask vector has the length of the whole training data, which is then sampled based on the character of the individual. Nonetheless, the whole, unsampled training set is employed to calculate the quality metrics. The final solution for the optimisation is applied to all the training data.

3. Lastly, the most complicated processing is applied for the 5x2 cross-validation protocol. Individuals have the length of the whole training data, but for each split, the sampling mask is only applied for the corresponding objects from the train set, while the validation set is not sampled. This means that, similarly to the case of testing on training set, each sample will always be used for testing; however, it might not be used for training. As in the standard 5x2 cross-validation, the split ratio is 50:50, and the process is repeated 5 times with training and validation sets swapping places. Finally, the sampling mask selected via optimisation is applied to the whole training data.

Each of the data processing is presented in Fig. 5.2.



(a) *Hold-out protocol*

(b) *Testing on training set protocol*

(c) *cross-validation protocol*

Figure 5.2: *Data processing on different protocols*

5.3 Experimental study

5.3.1 Objectives

The study, experiments and results analysis were conducted to find the answers to the following questions:

RQ1 Is there a difference in the quality of the Pareto front obtained by different estimation protocols?

RQ2 Which protocol estimates the actual objective values the best?

RQ3 Solution selection based on which protocol results in the classifiers of the best quality?

5.3.2 Setup

Choice of benchmark datasets. The experiments are conducted using 39 benchmark datasets from the KEEL [175] and UCI [176] repositories. All of the datasets represent binary problems. The datasets vary in size (both in samples and dimensionality) along with different imbalance ratios (ranging from 1.87 to 82.00). Descriptions of the used datasets can be found in the Table 5.1.

Parameter setting. Most of the parameters stemmed from the NSGA II optimisation algorithm - the size of population was 200, the number of iterations was 500, and the chosen operators were two-point crossover and flip mutation. As for the protocols themselves, the holdout split was 80% of the samples for the training and 20% for validation, and the cross-validation split was 50-50 and was repeated 5 times.

Solution choice rules. In the case of experiments 1 and 2, whole Pareto front estimations were used to calculate metrics. For the last experiment, three solutions from each optimisation result were selected - one with the best precision score, one with the best recall score and the one where the difference between both objectives was the smallest (labelled *balanced*).

Tested classifier Two classifiers were employed in this research, both in calculating optimisation objectives' values and final assessment:

1. Decision Tree Classifier (CART) - with *Gini impurity* as split criterion, no max depth set, miniminum samples split equal to 2 and minimum samples leaf equal to 1,

Table 5.1: *Description of datasets. #S denotes the number of samples, #F stands for the number of features, and IR indicates the imbalance ratio*

DATASET	#S	#F	IR [%]	dataset	#S	#F	IR [%]
page-blocks0	2736	9	8.79	*adult*	16280	14	3.15
bank additional	20594	20	7.88	*glass1*	107	9	1.82
glass0	107	9	2.06	*ecoli-0-6-7 vs 5*	110	6	10.00
ecoli-0-6-7 vs 3-5	111	7	9.09	*ecoli-0-2-6-7 vs 3-5*	112	7	9.18
ecoli-0-1 vs 2-3-5	122	7	9.17	*haberman*	153	3	2.78
ecoli1	168	7	3.36	*ecoli2*	168	7	5.46
ecoli3	168	7	8.60	*ecoli-0-1-4-7 vs 2-3-5-6*	168	7	10.59
yeast-1 vs 7	229	7	14.30	*page-blocks-1-3 vs 4*	236	10	15.86
yeast-2 vs 8	241	8	23.10	*yeast-0-3-5-9 vs 7-8*	253	8	9.12
yeast-2 vs 4	257	8	9.08	*yeast-0-5-6-7-9 vs 4*	264	8	9.35
yeast-1-4-5-8 vs 7	346	8	22.10	*pima*	384	8	1.87
winequality-white-3 vs 7	450	11	44.00	*yeast-1-2-8-9 vs 7*	473	8	30.57
yeast-0-2-5-6 vs 3-7-8-9	502	8	9.14	*flare-F*	533	11	23.79
kr-vs-k-zero vs eight	730	6	53.07	*poker-8 vs 6*	738	10	85.88
winequality-white-3-9 vs 5	741	11	58.28	*yeast3*	742	8	8.10
poker-8-9 vs 6	742	10	58.40	*yeast6*	742	8	41.40
yeast5	742	8	32.73	*yeast4*	742	8	28.10
winequality-red-4	799	11	29.17	*abalone-19 vs 10-11-12-13*	811	8	49.69
abalone-20 vs 8-9-10	958	8	72.69	*poker-8-9 vs 5*	1037	10	82.00
abalone-17 vs 7-8-9-10	1169	8	39.31				

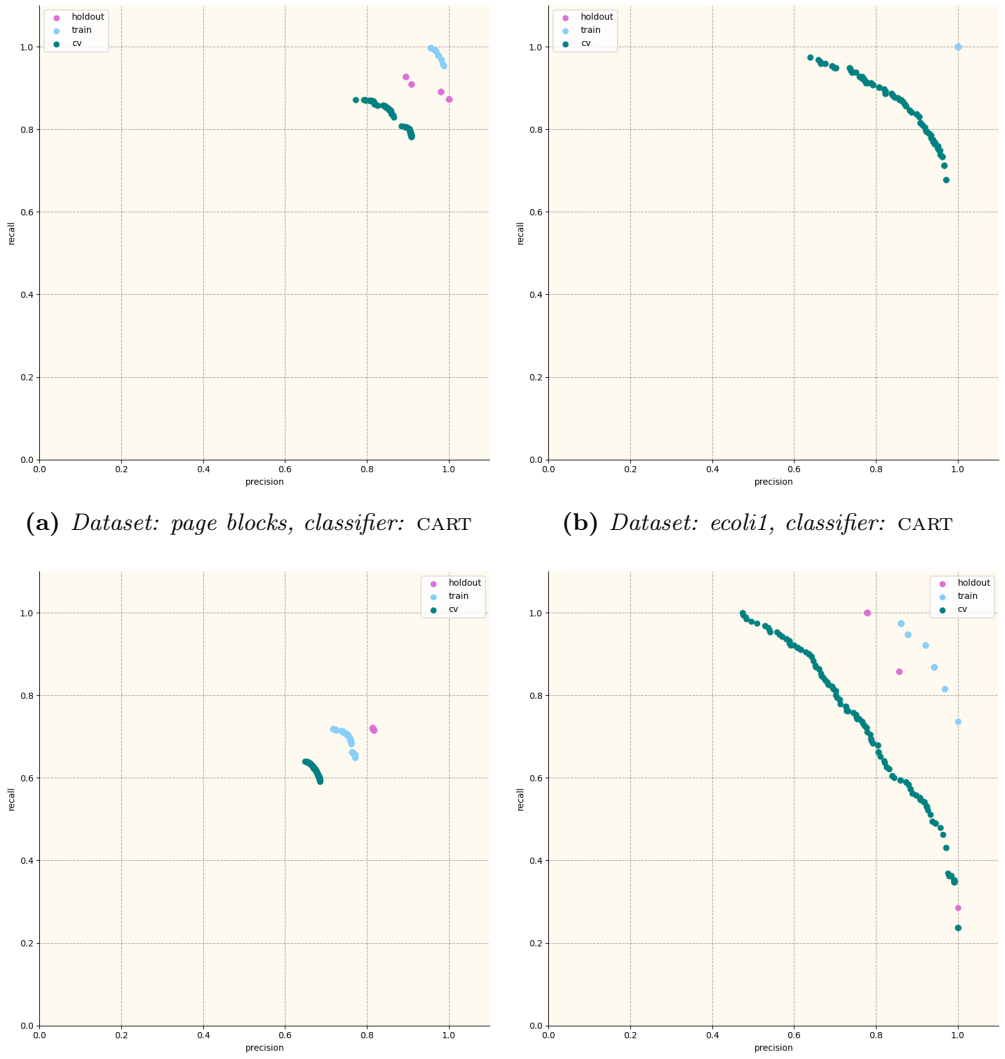
2. KNN Classifier with $k = 5$.

5.3.3 Results

5.3.3.1 Pareto fronts quality

The first experiment was dedicated to researching the difference between the quality of the Pareto fronts obtained with different assessment protocols. The examples of result sets are presented in Figure 5.3.

The first observation is that solutions obtained with holdout and test on train methodology significantly dominate the models assessed with cross-validation. Their sets are also more compact and thus less diverse, which is especially noticeable in the case of smaller datasets (*ecoli1* and *glass1*). Although technically worse in terms of prediction quality, the cross-validation protocol results in more unique solutions (in the context of varying pairs of objectives), when very often hold-out and test on train methodologies diverged into one, albeit perfect, solution.



(a) *Dataset: page blocks, classifier:* CART

(b) *Dataset: ecoli1, classifier:* CART

(c) *Dataset: adult, classifier:* KNN

(d) *Dataset: glass1, classifier:* KNN

Figure 5.3: *Example of Pareto fronts obtained with different objectives estimation method*

Tables 5.2 and 5.3 present the average number of unique solutions and the mean maximum spread, being the biggest distance between solutions in the Pareto front. It may be noticed that the hold-out method results in very few different solutions. However, it sometimes has the biggest spread amongst the tested protocols. The testing on training set usually generates a more diverse result set, although it still very often ends with only one solution with the highest objective score. The cross-validation protocol almost always creates at least 10 different solutions, which often go in pairs with their wide range.

Table 5.2: *Pareto fronts assessment employing* CART *for different objectives calculation protocols*

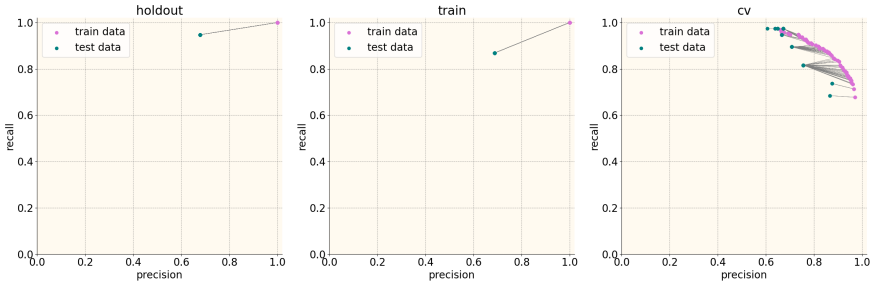
DATASET	hold-out		train		cv	
	unique solutions	max spread	unique solutions	max spread	unique solutions	max spread
adult	24.8	**0.067**	75.6	0.060	**132.2**	0.051
bank_additional	15.8	**0.095**	**69.9**	0.074	69.4	0.059
page-blocks0	5.8	0.097	17.9	0.062	**123.8**	**0.137**
glass1	1.5	0.038	1.0	0.000	**102.2**	**0.425**
glass0	2.1	0.108	1.0	0.000	**82.0**	**0.369**
ecoli-0-6-7_vs_5	1.0	0.000	1.0	0.000	**14.1**	**0.327**
ecoli-0-6-7_vs_3-5	1.0	0.000	1.0	0.000	**13.6**	**0.280**
ecoli-0-2-6-7_vs_3-5	1.1	0.050	1.0	0.000	**20.6**	**0.300**
ecoli-0-1_vs_2-3-5	1.0	0.000	1.0	0.000	**32.5**	**0.446**
haberman	4.1	0.227	3.3	0.041	**138.8**	**0.434**
ecoli1	1.4	0.039	1.5	0.016	**98.4**	**0.366**
ecoli2	1.4	0.100	1.0	0.000	**43.0**	**0.281**
ecoli3	1.2	0.040	1.0	0.000	**67.0**	**0.475**
ecoli-0-1-4-7_vs_2-3-5-6	1.2	0.067	1.2	0.007	**32.2**	**0.461**
yeast-1_vs_7	2.2	0.317	1.0	0.000	**45.2**	**0.455**
page-blocks-1-3_vs_4	1.0	0.000	1.0	0.000	**7.3**	**0.242**
yeast-2_vs_8	1.2	0.117	1.2	0.010	**12.8**	**0.380**
yeast-0-3-5-9_vs_7-8	2.4	0.238	1.9	0.044	**87.6**	**0.545**
yeast-2_vs_4	1.6	0.093	1.0	0.000	**65.8**	**0.416**
yeast-0-5-6-7-9_vs_4	1.5	0.089	2.0	0.032	**75.6**	**0.416**
yeast-1-4-5-8_vs_7	2.7	**0.440**	1.6	0.020	**31.8**	0.330
pima	6.7	0.211	17.7	0.158	**146.8**	**0.232**
winequality-white-3_vs_7	1.1	0.050	1.0	0.000	**15.6**	**0.482**
yeast-1-2-8-9_vs_7	2.0	0.310	1.7	0.050	**33.4**	**0.364**
yeast-0-2-5-6_vs_3-7-8-9	3.7	0.261	8.5	0.163	**76.0**	**0.293**
flare-F	2.7	0.217	12.6	0.416	**67.0**	**0.482**
kr-vs-k-zero_vs_eight	1.0	0.000	1.0	0.000	**22.1**	**0.321**
poker-8_vs_6	1.1	0.050	1.2	0.011	**3.4**	**0.342**
winequality-white-3-9_vs_5	1.8	0.172	1.0	0.000	**26.2**	**0.378**
yeast3	2.3	0.084	4.3	0.045	**124.8**	**0.245**
poker-8-9_vs_6	1.1	0.050	1.0	0.000	**6.8**	**0.527**
yeast6	1.7	0.177	1.2	0.011	**39.6**	**0.339**
yeast5	1.0	0.000	1.0	0.000	**33.2**	**0.304**
yeast4	1.8	0.162	1.6	0.015	**66.0**	**0.417**
winequality-red-4	3.0	0.383	3.5	0.109	**54.8**	**0.440**
abalone-19_vs_10-11-12-13	2.4	**0.550**	3.9	0.185	**11.6**	0.206
abalone-20_vs_8-9-10	2.3	0.320	2.5	0.085	**22.4**	**0.352**
poker-8-9_vs_5	1.2	0.067	1.8	0.042	**17.8**	**0.316**
abalone-17_vs_7-8-9-10	3.1	**0.343**	7.4	0.219	**39.8**	0.317

Table 5.3: *Pareto fronts assessment employing* KNN *for different objectives calculation protocols*

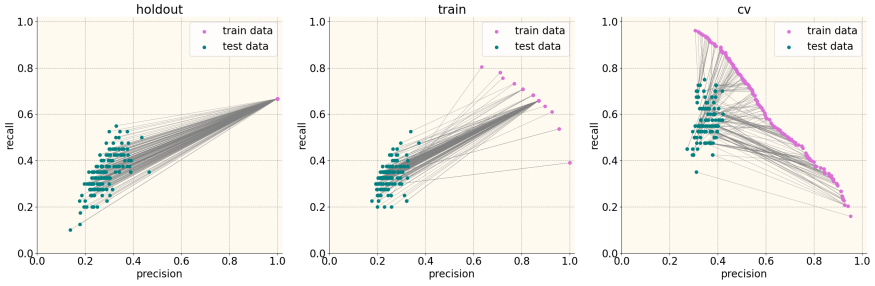
DATASET	hold-out		train		cv	
	unique solutions	max spread	unique solutions	max spread	unique solutions	max spread
adult	15.0	0.105	182.3	**0.124**	**343.2**	0.097
bank_additional	15.8	**0.095**	**69.9**	0.074	69.4	0.059
page-blocks0	3.4	0.156	29.9	0.205	**230.4**	**0.322**
glass1	2.1	0.167	11.2	0.244	**199.8**	**0.682**
glass0	2.4	0.111	11.8	0.306	**211.0**	**0.647**
ecoli-0-6-7_vs_5	1.0	0.000	2.2	0.160	**9.8**	**0.260**
ecoli-0-6-7_vs_3-5	1.0	0.000	2.1	0.209	**11.3**	**0.294**
ecoli-0-2-6-7_vs_3-5	1.0	0.000	2.1	0.227	**11.5**	**0.327**
ecoli-0-1_vs_2-3-5	1.2	0.100	2.0	0.175	**2.4**	**0.215**
haberman	2.0	0.188	18.3	0.538	**262.4**	**0.796**
ecoli1	1.7	0.149	7.3	0.177	**92.2**	**0.412**
ecoli2	1.5	0.140	2.7	0.086	**29.0**	**0.285**
ecoli3	1.1	0.050	8.2	0.399	**60.1**	**0.468**
ecoli-0-1-4-7_vs_2-3-5-6	1.5	0.167	3.5	0.209	**13.0**	**0.350**
yeast-1_vs_7	1.9	0.517	2.8	0.480	**15.4**	**0.658**
page-blocks-1-3_vs_4	2.0	0.333	2.1	0.086	**21.8**	**0.489**
yeast-2_vs_8	1.3	0.183	**2.8**	0.520	2.8	**0.544**
yeast-0-3-5-9_vs_7-8	2.0	0.440	5.2	0.544	**60.2**	**0.734**
yeast-2_vs_4	1.9	0.217	2.0	0.133	**15.2**	**0.274**
yeast-0-5-6-7-9_vs_4	1.7	0.260	4.8	0.412	**75.8**	**0.658**
yeast-1-4-5-8_vs_7	1.6	0.420	2.6	**0.700**	**4.8**	0.422
pima	2.6	0.144	34.8	0.289	**320.4**	**0.483**
winequality-white-3_vs_7	1.1	0.067	**1.9**	**0.770**	1.4	0.070
yeast-1-2-8-9_vs_7	1.2	0.138	**2.2**	**0.660**	2.1	0.395
yeast-0-2-5-6_vs_3-7-8-9	3.1	0.346	8.4	0.432	**96.8**	**0.667**
flare-F	2.2	0.493	12.2	0.670	**50.2**	**0.690**
kr-vs-k-zero_vs_eight	1.5	0.250	2.1	0.185	**11.2**	**0.408**
poker-8_vs_6	1.1	0.050	**2.0**	**0.504**	1.9	0.219
winequality-white-3-9_vs_5	1.3	0.192	**2.0**	**0.667**	1.8	0.173
yeast3	1.8	0.089	8.2	0.192	**74.4**	**0.280**
poker-8-9_vs_6	1.1	0.050	**2.0**	0.392	2.0	**0.466**
yeast6	1.7	0.249	4.6	0.389	**42.8**	**0.535**
yeast5	1.3	0.065	6.6	0.273	**69.8**	**0.624**
yeast4	1.6	0.340	4.4	0.546	**52.8**	**0.691**
winequality-red-4	1.6	0.401	3.6	**0.777**	**9.0**	0.499
abalone-19_vs_10-11-12-13	1.0	0.000	**1.9**	**0.760**	1.4	0.070
abalone-20_vs_8-9-10	1.3	0.190	**2.8**	**0.759**	1.6	0.108
poker-8-9_vs_5	1.0	0.000	**1.6**	**0.553**	1.2	0.017
abalone-17_vs_7-8-9-10	1.7	0.480	2.6	0.679	**6.0**	**0.716**

5.3.3.2 Final quality estimation

The previous experiment showed that the holdout and test on train protocols obtain better objective values and thus prediction quality. However, estimations on training sets are unreliable, and the hold-out validation set, in some cases, is too small and has too large a variance. Therefore, **RQ2** needs to be answered to determine if it is possible to rely on the criteria score obtained during optimisation. To achieve that, the distances between the optimisation solutions and respective objectives - recall and precision - calculated on a separate test set, were computed. Tables 8.14 and 8.18 present the average distance values over each solution across every fold, while Figure 5.4 shows the examples of such calculations.



(a) *Dataset: ecoli1, classifier: decision tree*



(b) *Dataset: haberman, classifier: knn*

Figure 5.4: *Example of distances of obtained results and ther respective test scores for each assessed protocol*

Visualisations show the significant difference between the quality of predicting data not used in the optimisation process. Though the bias is to be expected, the biggest problem is that the range of test scores can be pretty wide, and the properties of specific solutions are uncertain. Very often, the dominance does not transfer to test data, and the sequence of the solutions (according to the single objective) also does not hold. Moreover, one pair of criteria values might respond to many different final scores, which is especially evident in the case of the holdout protocol. It seems better for the cross-validation protocol, though sometimes several optimisation results respond to a single point on the test data.

Table 5.4: *Average distance from train to test Pareto front (*CART*)*

DATASET	holdout	train	cv
adult	0.129	0.321	**0.026**
bank_additional	0.167	0.409	**0.052**
page-blocks0	0.203	0.246	**0.060**
glass1	0.553	0.506	**0.183**
glass0	0.435	0.432	**0.165**
ecoli-0-6-7_vs_5	0.421	0.416	**0.212**
ecoli-0-6-7_vs_3-5	0.474	0.481	**0.253**
ecoli-0-2-6-7_vs_3-5	0.540	0.486	**0.282**
ecoli-0-1_vs_2-3-5	0.583	0.526	**0.335**
haberman	0.818	0.878	**0.300**
ecoli1	0.336	0.380	**0.138**
ecoli2	0.372	0.397	**0.175**
ecoli3	0.602	0.645	**0.231**
ecoli-0-1-4-7_vs_2-3-5-6	0.617	0.520	**0.277**
yeast-1_vs_7	0.816	0.914	**0.418**
page-blocks-1-3_vs_4	0.271	0.138	**0.131**
yeast-2_vs_8	0.557	0.702	**0.298**
yeast-0-3-5-9_vs_7-8	0.825	0.975	**0.269**
yeast-2_vs_4	0.427	0.447	**0.252**
yeast-0-5-6-7-9_vs_4	0.749	0.864	**0.281**
yeast-1-4-5-8_vs_7	0.953	1.241	**0.436**
pima	0.509	0.539	**0.150**
winequality-white-3_vs_7	0.830	1.083	**0.416**
yeast-1-2-8-9_vs_7	0.547	1.015	**0.367**
yeast-0-2-5-6_vs_3-7-8-9	0.578	0.642	**0.181**
flare-F	0.968	0.839	**0.392**
kr-vs-k-zero_vs_eight	0.394	**0.131**	0.239
poker-8_vs_6	0.591	0.417	**0.412**
winequality-white-3-9_vs_5	0.903	1.086	**0.404**
yeast3	0.347	0.466	**0.100**
poker-8-9_vs_6	0.637	0.505	**0.270**
yeast6	0.871	0.819	**0.340**
yeast5	0.549	0.582	**0.217**
yeast4	0.935	1.000	**0.315**
winequality-red-4	0.995	1.179	**0.414**
abalone-19_vs_10-11-12-13	0.982	1.155	**0.303**
abalone-20_vs_8-9-10	0.980	0.977	**0.378**
poker-8-9_vs_5	1.138	1.157	**0.372**
abalone-17_vs_7-8-9-10	0.867	0.928	**0.298**

Table 5.5: *Average distance from train to test Pareto front (*KNN*)*

DATASET	holdout	train	cv
adult	0.226	0.157	**0.033**
bank_additional	0.293	0.491	**0.190**
page-blocks0	0.183	0.148	**0.075**
glass1	0.502	0.367	**0.176**
glass0	0.510	0.367	**0.190**
ecoli-0-6-7_vs_5	0.600	**0.239**	0.280
ecoli-0-6-7_vs_3-5	0.502	**0.266**	0.283
ecoli-0-2-6-7_vs_3-5	0.519	**0.261**	0.269
ecoli-0-1_vs_2-3-5	0.352	**0.190**	0.193
haberman	0.786	0.588	**0.343**
ecoli1	0.337	0.281	**0.172**
ecoli2	0.270	0.179	**0.134**
ecoli3	0.652	0.370	**0.233**
ecoli-0-1-4-7_vs_2-3-5-6	0.322	**0.208**	0.225
yeast-1_vs_7	0.521	0.413	**0.386**
page-blocks-1-3_vs_4	0.414	**0.279**	0.308
yeast-2_vs_8	0.416	**0.236**	0.286
yeast-0-3-5-9_vs_7-8	0.496	0.490	**0.375**
yeast-2_vs_4	0.220	0.248	**0.201**
yeast-0-5-6-7-9_vs_4	0.640	0.571	**0.419**
yeast-1-4-5-8_vs_7	0.449	0.892	**0.424**
pima	0.482	0.422	**0.187**
winequality-white-3_vs_7	0.105	0.911	**0.092**
yeast-1-2-8-9_vs_7	0.326	0.553	**0.276**
yeast-0-2-5-6_vs_3-7-8-9	0.431	0.307	**0.229**
flare-F	0.809	0.647	**0.450**
kr-vs-k-zero_vs_eight	0.435	**0.348**	0.352
poker-8_vs_6	**0.281**	0.397	0.390
winequality-white-3-9_vs_5	0.320	0.684	**0.210**
yeast3	0.284	0.263	**0.167**
poker-8-9_vs_6	0.399	0.391	**0.345**
yeast6	0.531	0.478	**0.366**
yeast5	0.586	0.444	**0.257**
yeast4	0.719	0.538	**0.429**
winequality-red-4	0.387	0.805	**0.340**
abalone-19_vs_10-11-12-13	**0.000**	0.874	0.081
abalone-20_vs_8-9-10	0.258	0.689	**0.201**
poker-8-9_vs_5	**0.000**	0.602	0.020
abalone-17_vs_7-8-9-10	0.679	0.662	**0.477**

It can be noticed in tables 8.14 and 8.18 that the cross-validation protocol has the lowest average distance from train to test score for every but one dataset in the case of the CART classifier. For the KNN model, the scores obtained with test on train are sometimes closer. Nevertheless, cross-validation estimations are better for most of the datasets, and when not, its performance is still close to the best. As for the hold-out and testing on training set protocols, there is no clear pattern with the first being better with some examples and the second with the others. It has to be highlighted that some average distances are really high, around 1, indicating significant overfitting of the classifier.

5.3.3.3 Classifiers performance

Lastly, even if the estimations are not accurate and overly optimistic, in the end, one of the most important aspects is the quality of the model's prediction. So to answer RQ3, the third experiment compares the quality of classifiers obtained by selecting three solutions for each protocol. The detailed results can be found in Tables 8.12 - 8.19 in the Appendix, while the summary and statistical evaluation can be seen in Figures 5.5 - 5.8.

The results show that, in general, solutions optimised via cross-validation obtained higher metric scores, followed by testing on training set and hold-out being the worst. There is also more significant diversification in cross-validation solutions, as the models selected based on each objective have the highest values of *precision* and *recall*, respectively, while ranking last in the opposite measure. The difference is especially apparent for the CART classifier. The distinction is less significant in the case of testing on training set and hold-out. However, this might be due to the selection of one solution for each of the three instances, as optimisation diverged into a single point. The models based on cross-validation obtained the best results in all metrics in the case of the CART classifier, often being statistically better than most other estimators. It must be noticed that for almost all assessment measures, the cross-validation solution selected by the highest *recall* came first, though it was the worst in the case of *precision*. However, the balanced solution always obtained the second-best results. The situation is a little bit more interesting for the KNN estimator, since the advantage of cross-validation appears smaller, with testing on training set achieving comparable results. In both cases, however, the hold-out protocol is distinguished by the worst results. Nevertheless, there are examples of problems where each of the protocols seems superior, sometimes by a big margin.

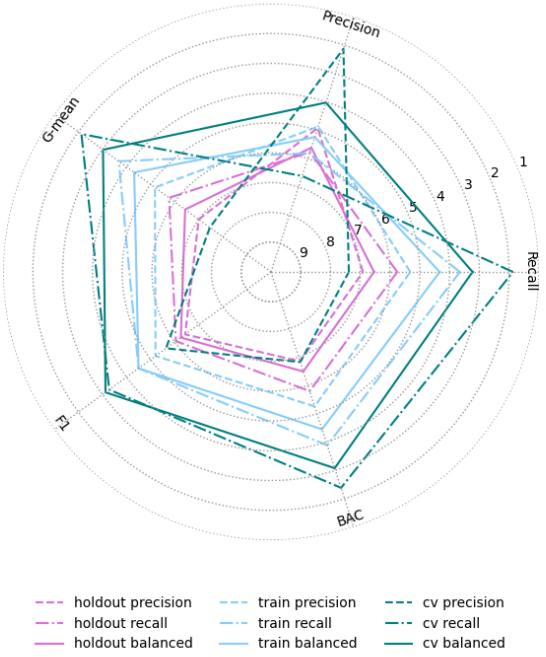
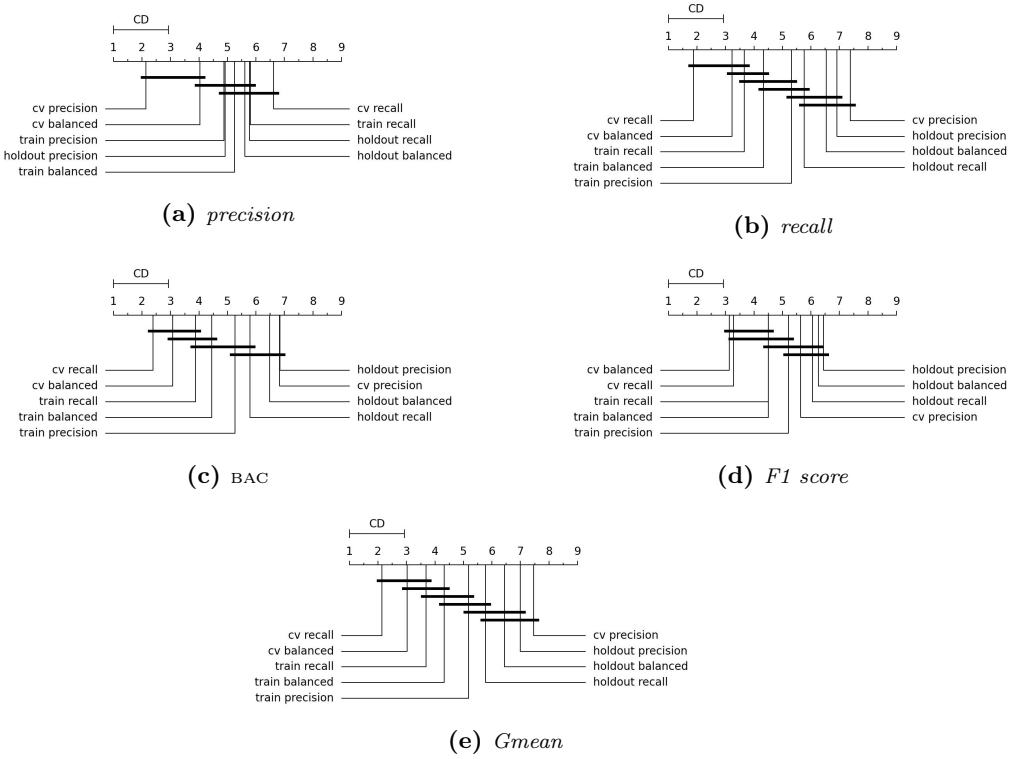


Figure 5.5: *Average rank of models based on solutions obtained from different objective calculation protocols (*CART*)*

Figure 5.6: *Average rank of models based on solutions obtained from different objective calculation protocols (*KNN*)*

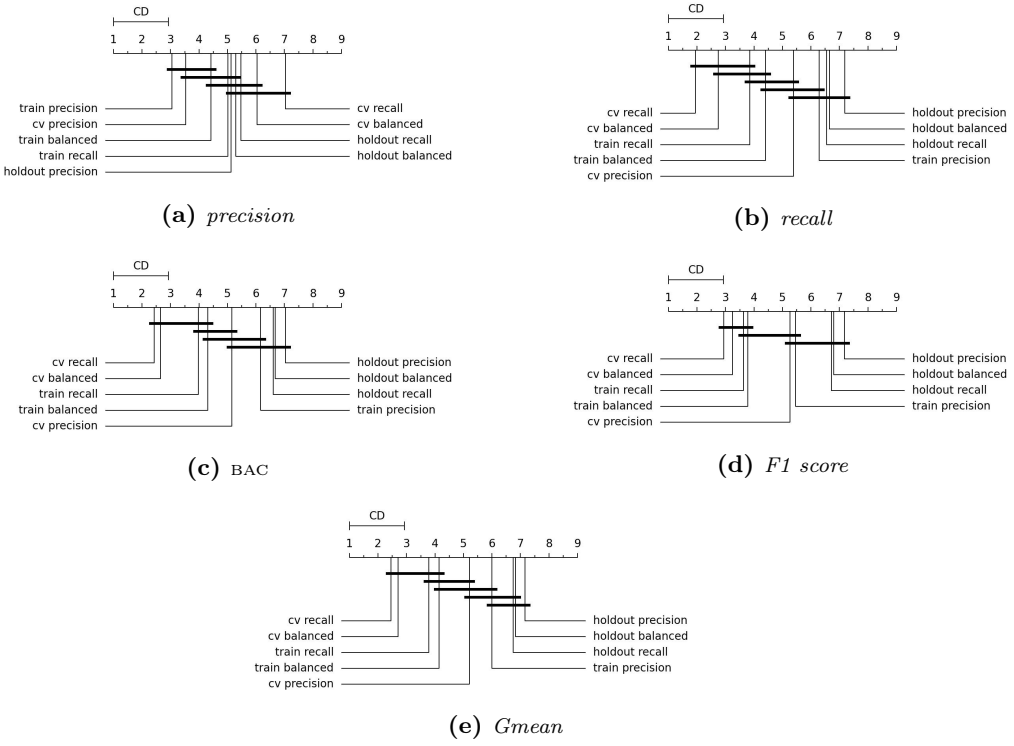


(a) *precision* **(b)** *recall*

(c) BAC **(d)** *F1 score*

(e) *Gmean*

Figure 5.7: *Results of Friedman and post hoc Nemenyi statistical tests for different metrics (*CART*)*



(a) *precision* **(b)** *recall*

(c) BAC **(d)** *F1 score*

(e) *Gmean*

Figure 5.8: *Results of Friedman and post hoc Nemenyi statistical tests for different metrics (*KNN*)*

5.4 Lessons learnt

In this chapter, the analysis of three different objective calculation protocols was conducted. The experiments were dedicated to assessing proposed methods in the context of the quality of generated Pareto fronts, estimation of the actual (test) scores and the prediction abilities. The first study showed the dominance of solutions obtained by hold-out and testing on training set protocols over cross-validation, while the latter resulted in wider and more numerous fronts. Second experiment determined that even if hold-out and testing on training set have a high training objective score, it might be due to overfitting, as criteria calculated on a separate test set were significantly different, with a high distance measure between respective points. Moreover, solutions with the same values of objectives would correspond to multiple significantly different scores in the test space. Cross-validation test solutions' scores were closer to the training counterparts. However, there was still a problem with test solutions not forming Pareto fronts and not corresponding directly to the order in train space. Lastly, the third analysis was dedicated to comparing particular solutions from each optimisation type. Results show that, in general, models based on cross-validation, especially those selected according to the highest *recall* and the smallest difference between criteria, achieve higher scores than the rest of the protocols. The difference was prominent in the case of the CART classifier. Nevertheless, for the KNN testing on training set also obtained close results. This might indicate that the properties of the classification algorithm have an influence on the optimisation and assessment approach.

It must be highlighted that there is a difference in computational complexity between each of the protocols, with hold-out being the fastest, testing on training computations taking a few times more time, since it employs a bigger validation set, and cross-validation being the slowest due to its repetitions. On this account, the fact that there was a dataset for each of the approaches, where it was superior, and the classification algorithm dependency, the user could consider adding protocol comparison to the development process. Moreover, depending on the type of employment of MOO in classifier training, it might be difficult or even impossible to utilise some protocols in objective calculation. For example, in the method proposed in Chapter 3, weights are being optimised for specific estimators trained on bootstrapped data. Hence, without some complex measures, it is infeasible to assess them with cross-validation (since it requires multiple training and testing).

Chapter 6

Surrogate criteria for gradient optimisation

This chapter answers the fourth research question - Is it possible to employ MOO gradient methods for the imbalanced data problem? *To accomplish this, a surrogate criteria will be proposed that substitutes popular objectives employed in metaheuristic algorithms - recall and specificity. The surrogate criteria will be evaluated utilising three different gradient optimisation methods* - MGDA, P*areto*MTL *and* COSMOS. *The proposed objectives will be assessed based on their similarity during training and final estimation of the Pareto front, as well as the overall quality of the resulting sets in the context of* MOO *and pattern recognition.*

6.1 Motivation

Previous research showed that metrics dedicated to assessing the recognition quality of the specific classes are appropriate as objectives for the MOO application in the imbalanced data task. Employing, for example, *recall* and *precision* in the parameter optimisation results in more balanced models with better generalisation capabilities. Nevertheless, calculating such criteria involves training a classifier, which may be very time-consuming depending on the selected algorithm and the data size. Utilisation of an estimation protocol based on repetition, such as cross-validation, additionally prolongs the process time. Moreover, the mechanisms in the popular genetic algorithms, such as mutations and crossovers, meant to improve the population and avoid converging into local optima, include randomness, generating many individuals that need assessment. While in general this methodology aims at finding potentially the best possible solution,

this also results in numerous costly computations, many of which do not factor into the final quality [186].

The solution to this problem could be employing a different type of optimisation method, for example, the ones that are characterised by a high rate of convergence. One of these groups of algorithms is gradient-based methods, usually employed in neural network training. There are several proposed multi-objective gradient algorithms that use different techniques to search in several directions, either by conducting optimisation from different starting points, segmenting the search space, or including a preferred direction vector in the training process. The problem with this approach is that previously used criteria cannot be utilised, as to calculate the gradient, the objective (loss) function needs to be differentiable, which metrics based on the confusion matrix are not. For this reason, the application of different optimisation criteria should be proposed, such that they would fit the imbalanced data problem and substitute for the class prediction quality measures in the sense of similar trends in training and final Pareto front estimations.

6.2 Proposed criteria

The most significant advantage of using criteria such as *precision*, *recall* (*true positive rate*), and *specificity* (*true negative rate*) is that their utilisation allows for the consideration of trade-offs of all classes' prediction quality. To emulate this feature in a gradient algorithm, the application of the *cross-entropy* loss function with weights enhancing individual classes was proposed.

$$l_n(x_n, y_n) = -w_{y_n} \log \frac{\exp(p_{n,y_n})}{\sum_{c=1}^{C} \exp(p_{n,c})} \tag{6.1}$$

where y_n is the class of the n-th sample, w_{y_n} is its weight, C is a number of classes and $p_{n,c}$ is a support of sample x_n belonging to the c-th class. To achieve the value for the whole learning set, a weight-oriented mean is used:

$$L(x, y) = \sum_{n=1}^{N} \frac{l_n}{\sum_{n=1}^{N} w_{y_n}} \tag{6.2}$$

The number of objectives of MOO equals the number of classes, and the weights differ for each criterion. For the i-th objective, weights take the form of:

$$w^i = [w_1^i, w_2^i, \ldots, w_C^i]^T$$

$$w_c^i = \begin{cases} 1 - \alpha & \text{if } c = i \\ \frac{\alpha}{C-1} & \text{if } c \neq i \end{cases}$$

The α parameter is meant to control the importance of the specific class recognition and the influence of the remaining classes.

In this research, only binary classification problems were considered. Hence, there were two objectives:

- **cross-entropy minority** - where the biggest weight was assigned to the minority class
- **cross-entropy majority** - analogically for the majority class.

6.3 Experimental study

To exhaustively test the proposed surrogate criteria, they were applied to three gradient-based algorithms - MGDA [142], ParetoMTL [143] and COSMOS [144], and evaluated primarily in terms of their consistency with *recall* and *specificity*, but also in the context of the quality of the Pareto front estimation and pattern recognition.

6.3.1 Objectives

The conducted set of experiments aims to prove the proposed surrogate criteria's utility by answering the following research questions:

RQ1 What is the influence of the α parameter on the properties of the surrogate objectives?

RQ2 What is the quality of the approximation of the Pareto fronts obtained utilising the proposed optimisation criteria?

RQ3 How well do solutions from surrogate space translate into the target space?

RQ4 How do optimisation methods using the proposed surrogate criteria perform compared to single-objective optimisation methods?

Table 6.1: *Description of datasets. #S denotes the number of samples, #F stands for the number of features, and IR indicates the imbalance ratio*

DATASET	#S	#F	IR [%]
adult	32561	14	31.71
page-blocks0	5472	9	11.38
bank_additional	41188	20	12.70
MiniBooNE_PID	130064	50	39.00
ecoli1	336	7	29.73
ecoli3	336	7	11.63
glass0	214	9	48.61
glass1	214	9	55.07
haberman	306	3	36.00
pima	768	8	53.60
yeast-0-2-5-6_vs_3-7-8-9	1004	8	10.94
yeast-0-3-5-9_vs_7-8	506	8	10.96
yeast-0-5-6-7-9_vs_4	528	8	10.69
yeast3	1484	8	12.24
yeast4	1484	8	3.55

6.3.2 Setup

Choice of benchmark datasets. Experiments were run on 15 datasets, which can be found on UCI [176] and KEEL [175] repositories. For analysis purposes, the datasets were divided into two categories: a) big datasets, with sizes exceeding 5000 samples, and b) small datasets, with sizes less than 5000 samples. The datasets also presented different imbalance ratio levels, i.e., the proportion between minority and majority class examples (from 3.55% to 55.07%)) and numbers of features (from 3 to 50). The data characteristic is presented in Table 6.1.

Experimental protocol. For comparison, each method used the same neural network architecture: three layers with rectified linear unit activation function and Adam optimiser with a learning rate of 0.001. The batches have sizes of 256 samples for big datasets and the whole data for small datasets. All experiments were conducted utilising a 5x2 cross-validation protocol with stratified data splits.

Implementation and reproducibility. The methods and evaluation were implemented using the Python programming language, modules such as *Pytorch* [187] and *scikit-learn* [177].

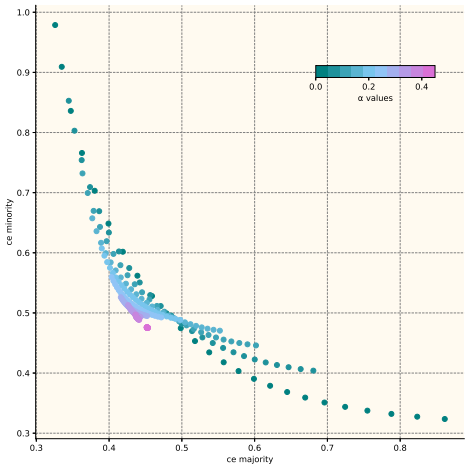
6.3.3 α hyperparameter analysis

To assess the influence of the α on the optimisation process, 10 values of the parameter ranging from 0.0 to 0.45 were tested on all datasets employing the COSMOS method. The range was selected due to the symmetric nature of both criteria. Resulting Pareto front estimations calculated on separate test sets (in both optimised - surrogate objectives space, as well as the target - *recall* and *specificity* space) were then compared and analysed. The example of such comparisons can be seen in Fig. 6.1.

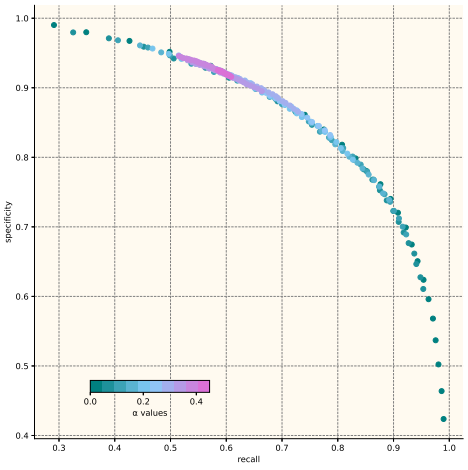
Table 6.2: *Average hypervolume measure for different α values in surrogate space*

DATASET	0.00	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45
adult	**0.378**	0.354	0.336	0.322	0.312	0.304	0.299	0.295	0.292	0.292
page-blocks0	**0.439**	0.415	0.414	0.416	0.417	0.420	0.422	0.421	0.422	0.420
bank_additional	**0.401**	0.363	0.357	0.351	0.349	0.344	0.346	0.350	0.354	0.357
MiniBooNE_PID	**0.434**	0.421	0.412	0.403	0.397	0.391	0.388	0.386	0.383	0.383
ecoli1	**0.395**	0.381	0.371	0.361	0.351	0.348	0.347	0.345	0.345	0.346
ecoli3	0.376	0.366	0.367	0.368	0.371	0.376	0.377	**0.379**	**0.379**	**0.379**
glass0	**0.331**	0.312	0.295	0.280	0.269	0.260	0.254	0.249	0.246	0.245
glass1	**0.295**	0.283	0.269	0.254	0.240	0.227	0.215	0.208	0.205	0.203
haberman	**0.255**	0.224	0.214	0.203	0.196	0.192	0.187	0.183	0.183	0.184
pima	**0.317**	0.298	0.280	0.262	0.243	0.227	0.213	0.208	0.203	0.202
yeast-0-2-5-6_vs_3-7-8-9	0.328	0.287	0.305	0.331	0.348	0.359	0.367	0.373	0.378	**0.380**
yeast-0-3-5-9_vs_7-8	0.257	0.237	0.263	0.291	0.314	0.331	0.342	0.349	0.356	**0.359**
yeast-0-5-6-7-9_vs_4	0.319	0.279	0.289	0.311	0.326	0.337	0.344	0.352	0.358	**0.361**
yeast3	**0.402**	0.381	0.382	0.386	0.390	0.395	0.397	0.399	0.401	**0.402**
yeast4	0.233	0.310	0.358	0.380	0.388	0.398	0.410	0.417	0.422	**0.425**

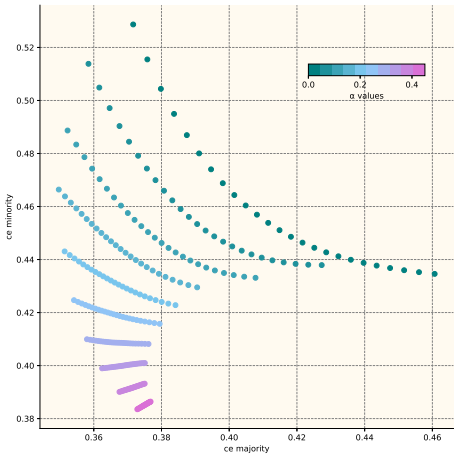
The most noticeable trend is that the smaller the α, and thus the more focus on a single class, the broader the resulting Pareto front is. With a higher α parameter, solutions are relatively compact. However, they also have better criteria values across all the results. *Hypervolume* values of surrogate (optimised) estimations of the Pareto front (Table 6.2) show that in the cases of some smaller datasets, the dominance is very prominent (as it significantly influences the scale of *hypervolume*). Nevertheless, in the case of bigger datasets (and some smaller ones), the dominance is not significant enough to translate into higher *hypervolume* scores. Moreover, this tendency seems not to cross to the target



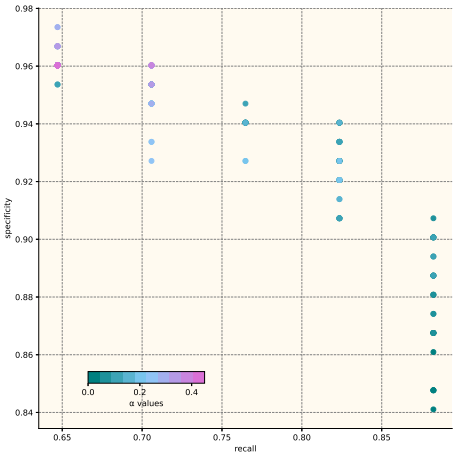
(a) *Resulting Pareto fronts in the surrogate objective space for different α values on adult test set*



(b) *Resulting Pareto fronts in the target objective space for different α values on adult test set*



(c) *Resulting Pareto fronts in the surrogate objective space for different α values on ecoli3 test set*



(d) *Resulting Pareto fronts in the target objective space for different α values on ecoli3 test set*

Figure 6.1: *Comparision of different α values*

recall-specificity space (Table 6.3), where solutions obtained from bigger α values do not dominate the rest. The other observation is that, together with the lower range of objective values, solutions obtained from higher α parameters tend to be slightly more biased towards majority class criteria, both in surrogate and target space.

As the goal of the multi-objective optimisation is to generate a wide set of solutions, which are furthermore not focused on one class recognition, for the rest of the experiments, $alpha = 0.0$ was selected. This also intuitively corresponds to the prerogatives of the target objectives, which only focus on the quality of one class recognition.

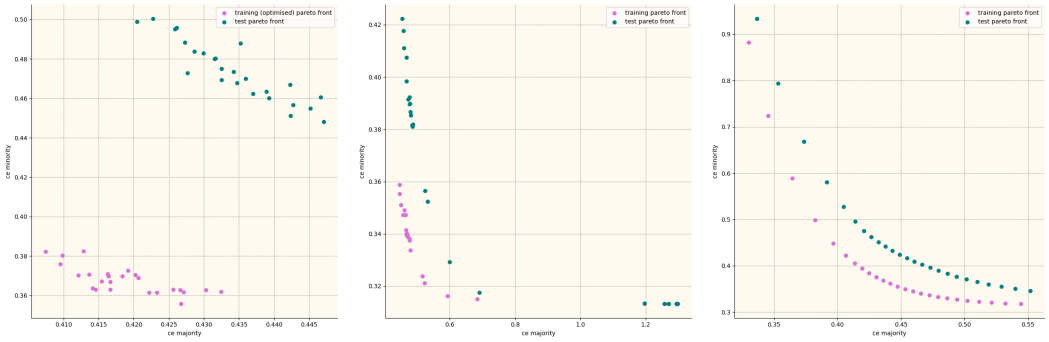
Table 6.3: *Average* hypervolume *measure for different* α *values in* recall-specificity *space*

DATASET	0.00	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45
adult	**0.886**	0.867	0.840	0.814	0.777	0.739	0.693	0.650	0.610	0.584
page-blocks0	**0.955**	0.915	0.896	0.868	0.844	0.826	0.805	0.772	0.736	0.700
bank_additional	**0.907**	0.844	0.809	0.750	0.686	0.599	0.548	0.513	0.490	0.472
MiniBooNE_PID	**0.953**	0.946	0.938	0.926	0.914	0.899	0.884	0.871	0.852	0.838
ecoli1	**0.888**	0.878	0.854	0.835	0.807	0.779	0.757	0.727	0.713	0.688
ecoli3	**0.849**	0.830	0.826	0.777	0.744	0.724	0.681	0.610	0.592	0.582
glass0	**0.803**	0.790	0.767	0.746	0.718	0.687	0.653	0.621	0.600	0.583
glass1	**0.763**	0.755	0.743	0.720	0.695	0.639	0.579	0.548	0.521	0.511
haberman	**0.665**	0.664	0.640	0.584	0.509	0.456	0.382	0.303	0.260	0.215
pima	**0.800**	0.785	0.761	0.730	0.684	0.632	0.583	0.554	0.520	0.498
yeast-0-2-5-6_vs_3-7-8-9	**0.793**	0.642	0.570	0.553	0.531	0.500	0.476	0.472	0.471	0.455
yeast-0-3-5-9_vs_7-8	**0.673**	0.572	0.476	0.442	0.429	0.371	0.295	0.248	0.228	0.212
yeast-0-5-6-7-9_vs_4	**0.778**	0.648	0.535	0.471	0.410	0.407	0.328	0.294	0.299	0.291
yeast3	**0.888**	0.846	0.811	0.783	0.758	0.750	0.725	0.710	0.700	0.693
yeast4	**0.722**	0.467	0.404	0.336	0.141	0.000	0.000	0.000	0.000	0.000

6.3.4 Pareto Front generation

To answer **RQ2**, the ability to generate Pareto fronts of the models utilising the proposed criteria was investigated. Firstly, the experiment was conducted to determine a relation between both objectives, as interconnected or dependent functions are not suited for the MOO process since they could result in a narrow Pareto front, in the worst case, reduced to a single point. Table 6.4 presents correlations of both criteria for all investigated methods. For big datasets, a strong negative correlation (usually above 0.90) may be noticed, meaning the objectives contradict each other and the optimisers can correctly approximate Pareto fronts. This is also true for most small datasets for COSMOS and ParetoMTL models. The negative correlation is not as strong in the case of MGDA, which is probably caused by its tendency to create narrower nondominated sets with dispersed dominated solutions due to the lack of data to train the model properly.

Secondly, it was assessed what the relationship is between training estimation Pareto fronts, where the optimisations have taken place, and the resulting Pareto fronts calculated on a separate test set. The example of such Pareto fronts can be seen in Fig. 6.2. Tables 6.7 and 6.8 present the average correlation between the proposed criteria on the training and test sets. The results showcase an almost ideal correlation in the case of ParetoMTL and COSMOS algorithms. The situation is more complex for the MGDA model,



(a) MGDA **(b)** PARETOMTL **(c)** COSMOS

Figure 6.2: *Examples of training and test Pareto fronts*

Table 6.4: *Average correlation of both criteria on testing data*

DATASET	MGDA	PARETOMTL	COSMOS
adult	-0.976	-0.953	-0.894
MiniBooNE_PID	-0.918	-0.977	-0.824
page-blocks0	-0.023	-0.382	-0.864
bank_additional	-0.956	-0.878	-0.878
ecoli1	-0.471	-0.822	-0.995
ecoli3	-0.415	-0.809	-0.953
glass0	-0.488	-0.914	-0.958
glass1	-0.736	-0.956	-0.946
haberman	-0.390	-0.977	-0.963
pima	-0.165	-0.881	-0.943
yeast-0-2-5-6_vs_3-7-8-9	-0.684	-0.962	-0.933
yeast-0-3-5-9_vs_7-8	-0.658	-0.975	-0.932
yeast-0-5-6-7-9_vs_4	-0.535	-0.872	-0.920
yeast3	-0.705	-0.852	-0.940
yeast4	-0.303	-0.925	-0.996

where there is a high correlation between train and test results for some datasets (notably the bigger examples), while for some, the relation seems not to have a very strong pattern. The reason for that could be the characteristics of the model, especially the fact that it does not learn to create a wide Pareto front. The algorithm tends to result in a denser set of solutions, where small changes might have a big impact on the correlation score. Nevertheless, other models demonstrate that usually there is a good translation between optimised criteria and their quality on unseen data, making it possible to select a solution from the Pareto front and precisely forecast its properties.

Next, the quality of the generated solution sets was investigated. Tables 6.9 and 6.10 show

Table 6.5: *Average correlation between training (optimised) and test majority weighted cross-entropy*

DATASET	MGDA	ParetoMTL	COSMOS
adult	0.987	1.000	1.000
MiniBooNE_PID	0.993	1.000	1.000
page-blocks0	0.617	1.000	0.999
bank_additional	0.988	1.000	1.000
ecoli1	0.292	0.995	0.994
ecoli3	0.635	0.998	0.999
glass0	0.172	0.984	0.999
glass1	0.311	0.993	0.999
haberman	0.543	0.998	0.999
pima	0.330	0.999	0.997
yeast-0-2-5-6_vs_3-7-8-9	0.836	1.000	0.998
yeast-0-3-5-9_vs_7-8	0.515	0.998	0.999
yeast-0-5-6-7-9_vs_4	0.913	0.998	0.999
yeast3	0.902	0.999	0.996
yeast4	0.830	1.000	1.000

Table 6.6: *Average correlation between training (optimised) and test minority weighted cross-entropy*

DATASET	MGDA	ParetoMTL	COSMOS
adult	0.939	0.990	1.000
MiniBooNE_PID	0.998	0.996	1.000
page-blocks0	-0.171	0.509	0.994
bank_additional	0.395	0.989	0.995
ecoli1	-0.698	0.972	0.984
ecoli3	-0.146	0.986	0.996
glass0	0.213	0.981	0.959
glass1	-0.154	0.990	0.995
haberman	0.211	0.991	0.999
pima	0.062	0.992	0.994
yeast-0-2-5-6_vs_3-7-8-9	0.617	0.989	0.998
yeast-0-3-5-9_vs_7-8	-0.224	0.985	0.990
yeast-0-5-6-7-9_vs_4	0.446	0.971	0.993
yeast3	-0.333	0.993	0.993
yeast4	0.037	0.942	1.000

proportions of nondominated solutions (Pareto front) on both training, on which optimisation was performed, and testing data. For both partitions, the COSMOS method results in almost entirely nondominated sets, which was to be expected because of its ability to learn generating Pareto fronts. In the case of algorithms without such a mechanism, the number of nondominated solutions is much smaller, around one-third for ParetoMTL and between 10 and 20 % for MGDA. Especially in the case of MGDA, there may be a difference in order of magnitude between proportions on small and big datasets, which might indicate difficulty learning from too few samples. Moreover, the MGDA generates much more limited Pareto fronts, resulting in a few nondominated solutions.

Although not all solutions form the Pareto front, dominated solutions could still be in its proximity (Fig. 6.3). Tables 6.11 and 6.12 present average distances of dominated points to the closest point in the Pareto front, normalised to the maximum Pareto front's values. For most datasets, the mean relative distance may not exceed 2%, proving that the solutions are primarily focused around Pareto fronts.

The above results show that the proposed criteria are suitable for multi-objective optimisation, creating diverse Pareto fronts. Nonetheless, the final quality of solutions is highly dependent on the employed optimisation algorithm, which is demonstrated primarily by comparing the resulting sets of methods with diversification mechanisms, such as COSMOS, to others like MGDA.

Table 6.7: *Average correlation between training (optimised) and test majority weighted cross-entropy*

DATASET	MGDA	ParetoMTL	COSMOS
adult	0.987	1.000	1.000
MiniBooNE_PID	0.993	1.000	1.000
page-blocks0	0.617	1.000	0.999
bank_additional	0.988	1.000	1.000
ecoli1	0.292	0.995	0.994
ecoli3	0.635	0.998	0.999
glass0	0.172	0.984	0.999
glass1	0.311	0.993	0.999
haberman	0.543	0.998	0.999
pima	0.330	0.999	0.997
yeast-0-2-5-6_vs_3-7-8-9	0.836	1.000	0.998
yeast-0-3-5-9_vs_7-8	0.515	0.998	0.999
yeast-0-5-6-7-9_vs_4	0.913	0.998	0.999
yeast3	0.902	0.999	0.996
yeast4	0.830	1.000	1.000

Table 6.8: *Average correlation between training (optimised) and test minority weighted cross-entropy*

DATASET	MGDA	ParetoMTL	COSMOS
adult	0.939	0.990	1.000
MiniBooNE_PID	0.998	0.996	1.000
page-blocks0	-0.171	0.509	0.994
bank_additional	0.395	0.989	0.995
ecoli1	-0.698	0.972	0.984
ecoli3	-0.146	0.986	0.996
glass0	0.213	0.981	0.959
glass1	-0.154	0.990	0.995
haberman	0.211	0.991	0.999
pima	0.062	0.992	0.994
yeast-0-2-5-6_vs_3-7-8-9	0.617	0.989	0.998
yeast-0-3-5-9_vs_7-8	-0.224	0.985	0.990
yeast-0-5-6-7-9_vs_4	0.446	0.971	0.993
yeast3	-0.333	0.993	0.993
yeast4	0.037	0.942	1.000

Table 6.9: *Average proportion [%] of nondominated solutions on training data*

DATASET	MGDA	ParetoMTL	COSMOS
adult	42.80	66.00	100.00
MiniBooNE_PID	42.80	43.60	100.00
page-blocks0	24.80	26.80	100.00
bank_additional	26.40	62.80	100.00
ecoli1	21.20	39.20	100.00
ecoli3	17.20	35.60	100.00
glass0	12.00	46.40	100.00
glass1	16.80	46.00	100.00
haberman	16.00	63.20	100.00
pima	17.60	44.80	100.00
yeast-0-2-5-6_vs_3-7-8-9	16.40	56.40	100.00
yeast-0-3-5-9_vs_7-8	10.80	52.00	100.00
yeast-0-5-6-7-9_vs_4	18.80	40.40	100.00
yeast3	13.20	38.40	100.00
yeast4	14.00	36.80	100.00

Table 6.10: *Average proportion [%] of nondominated solutions on testing data*

DATASET	MGDA	ParetoMTL	COSMOS
adult	45.60	57.20	100.00
page-blocks0	16.00	36.40	99.60
bank_additional	28.75	54.00	100.00
MiniBooNE_PID	50.00	40.80	98.40
ecoli1	17.60	32.00	100.00
ecoli3	12.80	17.20	98.80
glass0	16.80	42.40	100.00
glass1	16.80	40.40	100.00
haberman	26.00	41.60	100.00
pima	19.20	35.20	100.00
yeast-0-2-5-6_vs_3-7-8-9	17.20	13.20	96.80
yeast-0-3-5-9_vs_7-8	17.20	25.20	100.00
yeast-0-5-6-7-9_vs_4	19.60	31.20	100.00
yeast3	17.20	29.60	98.40
yeast4	21.60	41.20	100.00

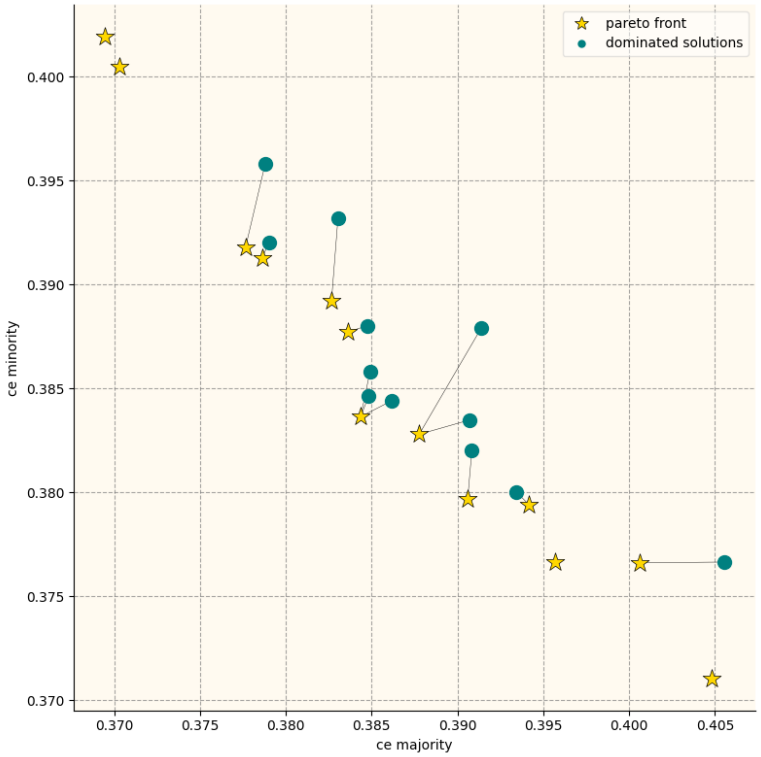


Figure 6.3: *Example of solutions obtained from* MGDA *algorithm with denoted Pareto front and distances to dominated points*

Table 6.11: *Average distance to Pareto front on training data*

DATASET	MGDA	ParetoMTL	COSMOS
adult	0.005	0.001	0.000
MiniBooNE_PID	0.003	0.000	0.000
page-blocks0	0.005	0.010	0.000
bank_additional	0.009	0.002	0.000
ecoli1	0.010	0.023	0.000
ecoli3	0.008	0.022	0.000
glass0	0.031	0.011	0.000
glass1	0.028	0.009	0.000
haberman	0.017	0.003	0.000
pima	0.019	0.009	0.000
yeast-0-2-5-6_vs_3-7-8-9	0.020	0.006	0.000
yeast-0-3-5-9_vs_7-8	0.036	0.007	0.000
yeast-0-5-6-7-9_vs_4	0.017	0.017	0.000
yeast3	0.008	0.022	0.000
yeast4	0.012	0.017	0.000

Table 6.12: *Average distance to Pareto front on testing data*

DATASET	MGDA	ParetoMTL	COSMOS
adult	0.003	0.001	0.000
MiniBooNE_PID	0.003	0.000	0.000
page-blocks0	0.004	0.011	0.000
bank_additional	0.003	0.001	0.000
ecoli1	0.014	0.021	0.000
ecoli3	0.009	0.023	0.000
glass0	0.018	0.012	0.000
glass1	0.033	0.008	0.000
haberman	0.010	0.003	0.000
pima	0.016	0.008	0.000
yeast-0-2-5-6_vs_3-7-8-9	0.013	0.006	0.000
yeast-0-3-5-9_vs_7-8	0.022	0.006	0.000
yeast-0-5-6-7-9_vs_4	0.023	0.017	0.000
yeast3	0.005	0.020	0.000
yeast4	0.012	0.012	0.000

6.3.5 Surrogate and target criteria relation

The aim of the second set of experiments was to answer **RQ3** - *How well do solutions from surrogate space translate into the target space* and thus investigate whether it is appropriate to employ the proposed criteria to emulate target objectives, namely *recall* and *specificity*, where *cross-entropy minority* corresponds to *recall*, as it focuses on minority class correct prediction, while analogically *cross-entropy majority* substitutes for *specificity*. Firstly, it was examined whether the behaviour of the surrogate functions matches the behaviour of the quality metrics, both during the training process and in the final Pareto front estimations. Tables 6.13 and 6.14 show the correlation between surrogate and target criteria during training, while 6.15 and 6.16 present their correlation in the resulting Pareto fronts. In the case of training, it has to be pointed out that for almost all datasets, there is a strong (> 0.7) correlation between corresponding objectives. This indicates that the proposed loss functions appropriately substitute for *recall* and *specificity*, as the improvement of surrogate criteria directly influences the progression of the target measure. The examples of weaker correlations were always related to smaller datasets, which could be attributed to fewer possible values of quality measures (meaning that the same value of recall would represent different values of *minority cross-entropy*).

As for the final solution sets correspondence, it may be noticed that for the big datasets, all three models have an almost perfect negative correlation, meaning that their course is almost exactly opposite, as surrogate ones need to be minimised and the targets maximised. The same could be said about the smaller datasets, especially in the case of the COSMOS algorithm, as in only a few examples, the correlation totals to less than 0.7. Moreover, it is worth noting that decreases in correlation values are connected to individual methods, as in all instances of a weaker relation, there is at least one other model with a correlation value above 0.9. These results indicate that the values and solutions placements are tightly related to the target space, meaning it is possible to select solutions aiming at specific target properties based on characteristics in the surrogate space.

The following experiment was intended to test whether the quality of the Pareto front in the surrogate space translates into the target space. For this purpose, it was investigated how many optimised Pareto front solutions are nondominated based on *recall* and *specificity* (Fig. 6.4). Tables 6.17 and 6.18 show the proportion of such cases compared to all solutions. Further, it was examined what the mean distance between solutions that are nondominated in surrogate space to the target space Pareto front is (Tables 6.19 and 6.20). Even though the solution is no longer part of the estimated Pareto front in the target space, it still could be placed sufficiently close to other nondominated points. Results indicate that not only are nondominated surrogate solutions close to the target

Table 6.13: *Average correlation of cross-entropy majority and specificity during training on test data*

DATASET	MGDA	ParetoMTL	COSMOS
adult	-0.939	-0.997	-0.999
MiniBooNE_PID	-0.997	-0.997	-0.978
page-blocks0	-0.726	-0.756	-0.880
bank_additional	-0.993	-0.999	-1.000
ecoli1	-0.603	-0.814	-0.081
ecoli3	-0.713	-0.766	0.056
glass0	-0.694	-0.673	-0.527
glass1	-0.808	-0.608	-0.702
haberman	-0.479	-0.560	-0.835
pima	-0.677	-0.639	-0.912
yeast-0-2-5-6_vs_3-7-8-9	-0.746	-0.752	-0.918
yeast-0-3-5-9_vs_7-8	-0.602	-0.700	-0.876
yeast-0-5-6-7-9_vs_4	-0.715	-0.764	-0.862
yeast3	-0.642	-0.852	-0.922
yeast4	-0.751	-0.771	-0.873

Table 6.14: *Average correlation of cross-entropy minority and recall during training on test data*

DATASET	MGDA	ParetoMTL	COSMOS
adult	-0.983	-0.962	-0.999
MiniBooNE_PID	-0.999	-1.000	-0.958
page-blocks0	-0.683	-0.880	-0.941
bank_additional	-0.997	-0.992	-0.999
ecoli1	-0.467	-0.663	-0.639
ecoli3	-0.296	-0.576	-0.505
glass0	-0.403	-0.623	-0.806
glass1	-0.312	-0.636	-0.822
haberman	-0.497	-0.637	-0.810
pima	-0.273	-0.648	-0.891
yeast-0-2-5-6_vs_3-7-8-9	-0.457	-0.691	-0.837
yeast-0-3-5-9_vs_7-8	-0.602	-0.705	-0.904
yeast-0-5-6-7-9_vs_4	-0.538	-0.650	-0.827
yeast3	-0.338	-0.635	-0.373
yeast4	-0.581	-0.584	-0.201

Table 6.15: *Correlation between majority weighted cross-entropy and specificity on train data*

DATASET	MGDA	ParetoMTL	COSMOS
adult	-0.997	-1.000	-1.000
MiniBooNE_PID	-0.998	NaN	-1.000
page-blocks0	-0.633	-0.984	-0.999
bank_additional	-0.999	-1.000	-1.000
ecoli1	-0.209	-0.896	-0.982
ecoli3	-0.586	-0.942	-0.974
glass0	-0.761	-0.867	-0.989
glass1	-0.942	-0.827	-0.986
haberman	-0.447	NaN	-0.995
pima	-0.733	-0.943	-0.999
yeast-0-2-5-6_vs_3-7-8-9	-0.619	-0.938	-0.995
yeast-0-3-5-9_vs_7-8	-0.838	-0.936	-0.999
yeast-0-5-6-7-9_vs_4	-0.859	-0.969	-0.997
yeast3	-0.788	-0.985	-0.995
yeast4	-0.921	-0.983	-0.998

Table 6.16: *Correlation between minority weighted cross-entropy and recall on test data*

DATASET	MGDA	ParetoMTL	COSMOS
adult	-0.996	-1.000	-1.000
MiniBooNE_PID	-0.998	NaN	-1.000
page-blocks0	-0.716	NaN	-0.995
bank_additional	-0.997	-1.000	-1.000
ecoli1	-0.793	-0.713	-0.950
ecoli3	-0.697	NaN	-0.875
glass0	-0.887	-0.181	-0.992
glass1	-0.919	-0.567	-0.990
haberman	-0.615	NaN	-0.990
pima	-0.728	-0.540	-0.995
yeast-0-2-5-6_vs_3-7-8-9	-0.567	-0.848	-0.989
yeast-0-3-5-9_vs_7-8	-0.783	-0.855	-0.984
yeast-0-5-6-7-9_vs_4	-0.890	-0.551	-0.992
yeast3	-0.625	-0.761	-0.978
yeast4	-0.883	-0.807	-0.681

Table 6.17: *Average proportion [%] of nondominated surrogate solutions which are also nondominated in target space (train data)*

DATASET	MGDA	ParetoMTL	COSMOS
adult	80.20	71.01	100.00
MiniBooNE_PID	85.38	100.00	100.00
page-blocks0	45.07	15.31	70.80
bank_additional	87.20	65.17	98.00
ecoli1	42.43	17.48	38.00
ecoli3	46.76	12.22	15.60
glass0	45.33	7.87	35.60
glass1	44.21	16.33	42.80
haberman	31.50	81.84	66.00
pima	44.74	31.72	84.80
yeast-0-2-5-6_vs_3-7-8-9	35.36	30.44	59.20
yeast-0-3-5-9_vs_7-8	38.33	12.86	41.20
yeast-0-5-6-7-9_vs_4	42.19	21.21	33.60
yeast3	36.00	23.37	35.20
yeast4	27.33	18.66	12.80

Table 6.18: *Average proportion [%] of nondominated surrogate solutions which are also nondominated in target space (test data)*

DATASET	MGDA	ParetoMTL	COSMOS
adult	80.15	73.41	100.00
MiniBooNE_PID	82.32	100.00	100.00
page-blocks0	47.95	21.20	79.20
bank_additional	89.00	67.91	100.00
ecoli1	32.10	27.20	36.00
ecoli3	38.11	17.31	19.60
glass0	34.38	17.83	45.60
glass1	63.33	18.36	53.20
haberman	16.96	90.09	72.00
pima	59.83	32.89	92.00
yeast-0-2-5-6_vs_3-7-8-9	30.94	36.82	58.40
yeast-0-3-5-9_vs_7-8	26.29	22.66	45.60
yeast-0-5-6-7-9_vs_4	45.33	28.55	45.60
yeast3	25.30	24.51	42.80
yeast4	35.56	26.65	12.80

Pareto fronts, but fairly often, they are also nondominated according to quality metrics, which means that Pareto fronts translate fairly well into the target space. It should be noted that the proportions are usually higher for the bigger datasets, especially for the ParetoMTL and COSMOS algorithms. Lower values for smaller data might be caused by the phenomenon of confusion matrix Pareto fronts being less diverse and sparse for smaller datasets, as small numbers of minority class samples result in fewer possible values of quality measures [180] and the tendency to overfitting.

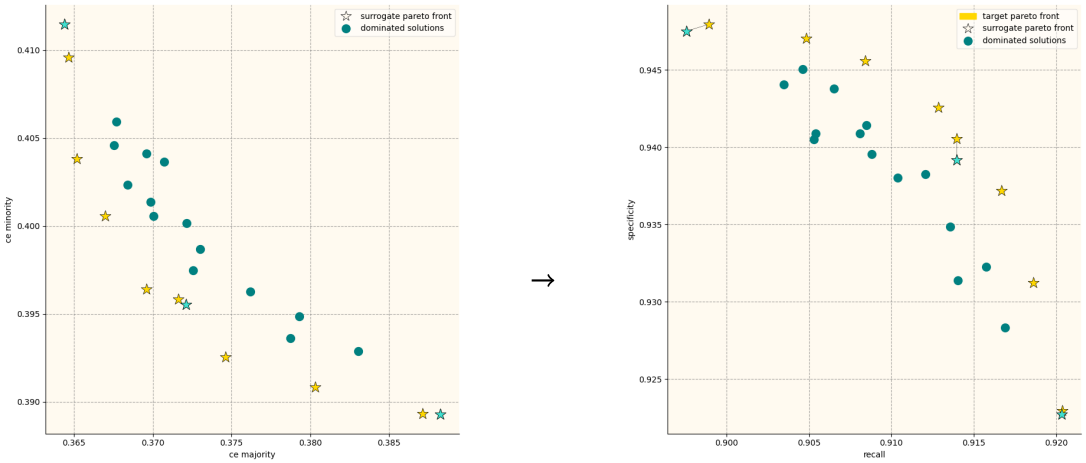
The obtained results show a strong relation between proposed cross-entropy-based criteria and desired quality measures, not only individually but also in multi-objective optimisation, which answers **RQ3**.

Table 6.19: *Average distance of nondominated surrogate solutions to the Pareto front in target space (train data)*

DATASET	MGDA	ParetoMTL	COSMOS
adult	0.001	0.025	0.000
MiniBooNE_PID	0.000	0.000	0.000
page-blocks0	0.002	0.095	0.003
bank_additional	0.000	0.046	0.000
ecoli1	0.005	0.058	0.011
ecoli3	0.004	0.074	0.018
glass0	0.011	0.086	0.034
glass1	0.015	0.062	0.032
haberman	0.015	0.007	0.011
pima	0.005	0.045	0.005
yeast-0-2-5-6_vs_3-7-8-9	0.006	0.054	0.014
yeast-0-3-5-9_vs_7-8	0.008	0.136	0.035
yeast-0-5-6-7-9_vs_4	0.006	0.087	0.023
yeast3	0.003	0.102	0.008
yeast4	0.005	0.088	0.025

Table 6.20: *Average distance of nondominated surrogate solutions to the Pareto front in target space (test data)*

DATASET	MGDA	ParetoMTL	COSMOS
adult	0.001	0.002	0.000
MiniBooNE_PID	0.000	0.000	0.000
page-blocks0	0.001	0.084	0.002
bank_additional	0.000	0.014	0.000
ecoli1	0.009	0.046	0.012
ecoli3	0.008	0.056	0.019
glass0	0.019	0.045	0.020
glass1	0.010	0.049	0.019
haberman	0.026	0.005	0.008
pima	0.005	0.053	0.002
yeast-0-2-5-6_vs_3-7-8-9	0.009	0.038	0.010
yeast-0-3-5-9_vs_7-8	0.021	0.061	0.018
yeast-0-5-6-7-9_vs_4	0.017	0.053	0.014
yeast3	0.005	0.071	0.005
yeast4	0.008	0.055	0.038



(a) *Solutions in surrogate space* **(b)** *Solutions in target space*

Figure 6.4: *Example of translation of nondominated solutions from surrogate to target space*

6.3.6 Single objective optimization comparison

Finally, the last study was conducted to answer **RQ4** - *How do optimisation methods using the proposed surrogate criteria perform compared to single-objective optimisation methods?*. The experiment was executed on all 15 datasets using six models with the same architecture but different objective functions to compare MOO methods using the proposed criteria with standard single-objective algorithms. For the single-objective problems, the following losses were used:

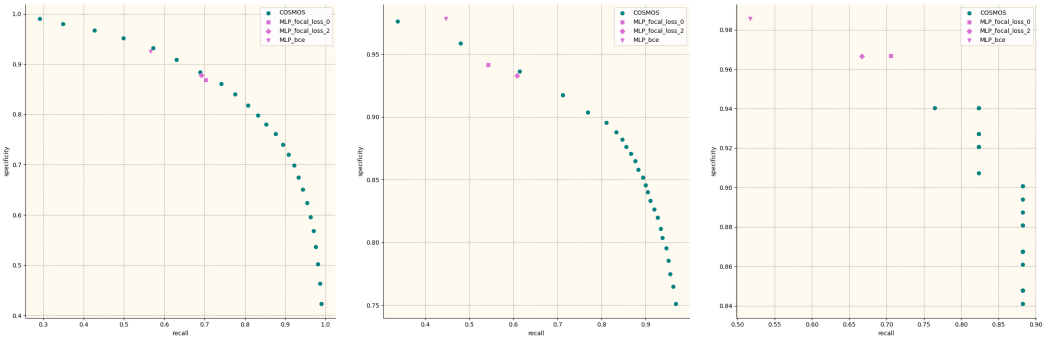
- *focal loss* function with two different γ parameter values ($\gamma = 0$ and $\gamma = 2$)
- *binary cross-entropy loss* with positive (minority) class weight inversely proportional to the ratio of minority class size to majority class size ($\frac{N_{maj}}{N_{min}}$)

As the COSMOS method presented the best ability to generate diverse Pareto fronts with the best translation from the surrogate to the target space, it was selected as the MOO representative. For clarity purposes, three solutions from the Pareto set will be compared with SOO methods:

- two extreme solutions from the Pareto front (COSMOS MIN and COSMOS MAX
- the solution from the middle of the Pareto front (COSMOS BAL)

The methods were assessed using previously employed imbalanced data performance measures - *precision*, *recall*, BAC, *F1 score*, and *Gmean*. The results were checked for statistical significance using the Friedman ranking test with the Nemenyi post hoc test.

As shown both in Figure 6.5 and Tables 8.25 - 8.24 (in the Appendix), very often the SOO methods were dominated by solutions from the Pareto front, and if not, there was at least one COSMOS solution better in the chosen metric. When the model optimised by a single criterion was superior according to one measure, there was always a different one where the MOO solutions were favourable. None of the compared models came first for all of the quality metrics. However, almost always, it was possible to choose a solution from a Pareto front with the best or close to the best performance, and sometimes the difference in quality was statistically significant (Fig. 6.7). Moreover, as presented in Fig. 6.6, utilising the MOO method with the proposed criteria allows the user to decide whether to select a solution favouring the preferred metrics or balancing all the measures. In contrast, the SOO solutions are usually biased towards one performance aspect (i.e., both *focal loss* functions lean towards *precision*). They are hard to control the direction of optimisation without excessive parameter testing. All of this proves the usability of the proposed criteria paired with adequate MOO method, thereby answering **RQ4**.



(a) *adult dataset* **(b)** *bank_ additional dataset* **(c)** *ecoli3 dataset*

Figure 6.5: *Comparison of* MOO *and* SOO *methods in recall and specificity space*

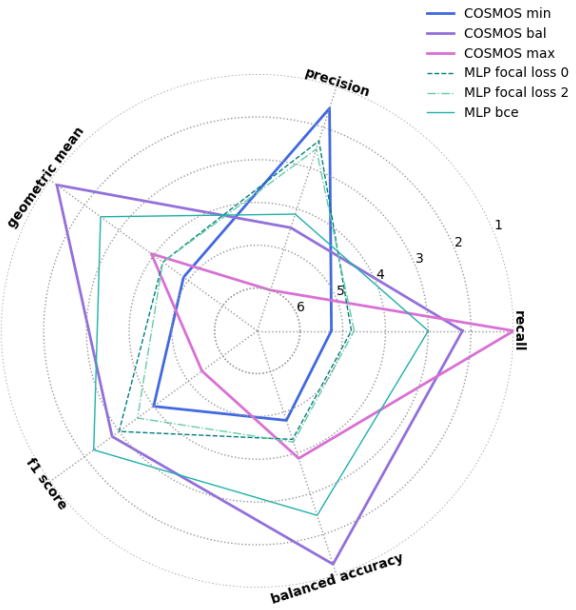
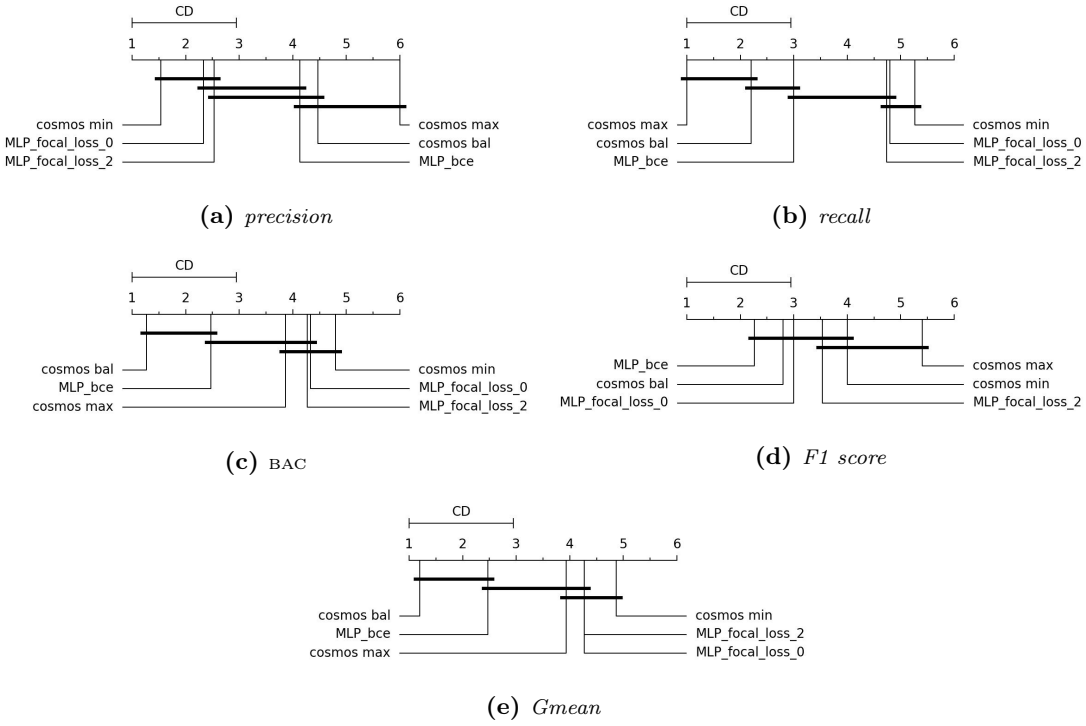


Figure 6.6: *Radar diagram of average ranks of the methods*



(a) *precision* **(b)** *recall*

(c) BAC **(d)** *F1 score*

(e) *Gmean*

Figure 6.7: *Critical difference diagrams base on Nemenyi post hoc test*

6.4 Lessons learnt

This chapter proposed the employment of weighted cross-entropy loss as a surrogate for recall and specificity quality metrics in multi-objective optimisation. The research was conducted to determine the best way of weighting the loss functions, investigate a relation, and thereby adequateness, between proposed and target criteria, as well as assess its usefulness in comparison to single objective gradient optimisation.

The first experiment showed that using smaller α, which translated to smaller (or even no) influence of the opposite class recognition, resulted in wider Pareto fronts, which were also less biased towards the majority class. The second experiment was conducted to assess the proposed loss functions as the criteria of the MOO algorithm. The outcome indicated that appropriately weighted cross entropies are not strictly dependent on each other, which consequently results in, depending on the algorithm, a non-dominated set of diverse solutions. Then it was determined that there is a strong correlation between optimised and test Pareto fronts, making it possible to select adequate solutions based on their training properties. The following experiment showcased the relationship between the surrogate and target criteria. The proposed objectives were highly correlated with the *recall* and *specificity* both during the training process, as well as the final Pareto fronts. The dominance relation between solutions also translated well from the surrogate and target spaces. These results showcase that it is feasible to train a neural network model aiming for good *recall* and *specificity* scores, but also to select the solutions based on their weighted cross-entropy loss values. Lastly, the experiment was conducted to compare the MOO model employing the proposed criteria to the model's training, utilising a single objective appropriate for imbalanced data learning. The results presented the advantage of employing multiple objectives, as they offer a broad selection of models with different properties, similar or better than the neural networks trained on single objective.

To conclude, the experimental evaluation proves that it is appropriate to employ weighted cross-entropy in place of recall and specificity in a multi-objective gradient optimisation algorithm, and it is beneficial for the imbalanced data classification problem compared to standard optimisation with a single loss function. Nevertheless, it must be noticed that this type of optimisation is limited to the methods using gradients for calculating parameters and thus is not model agnostic.

Chapter 7

Pareto front solutions analysis

This chapter considers the last research question on evaluating the diversity of classifiers that form the Pareto front. The explainable artificial intelligence technique is proposed to help the user make a conscious decision, i.e., choose the tailored solution. Considerations were limited to a pool of interpretable models, and the visualisations used to show the importance of individual features based on the location of classifiers in the Pareto front. The limitations of the proposed method is also discussed.

7.1 Motivation

Previous research demonstrated some difficulties with employing multi-objective optimisation in the imbalanced data problem, such as quality criteria estimation and computational complexity. However, there is also one challenge related intrinsically to the nature of optimising according to multiple criteria, which is the selection of the final solution. The problem is not trivial even in the case of the typical multi-objective optimisation employment, where the criteria are deterministic. Multiple methods have been proposed to aid the user in making a decision[147]. Nevertheless, they might be insufficient in the case of an imbalanced pattern recognition task. Firstly, these algorithms usually assume that the decision maker can give particular preferences about each criterion's importance. However, it is often difficult to clearly determine the cost of class recognition. Moreover, the differences between solutions are sometimes minimal and hard to appraise without factors other than optimisation objectives. Secondly, because the quality metrics are estimated and not deterministic, there can always be some differences between the scores obtained by the optimisation algorithm and their real values. It is especially prominent in the case of imbalanced and small data, where estimations are limited, due to few

possible unique scores, and of high variance [21]. The results showed that the solution arrangement does not always directly transfer into the test space, meaning it is essential to consider alternatives to the assessment of solutions besides their criteria values.

7.2 Solution selection based on the interpretable model

The easiest method to analyse the models resulting from the MOO solutions is to employ a classifier that is interpretable by nature, such as decision trees. In chapter 3, the sampling method employing the CART was proposed. Since each solution represented parameters for the sampler, resulting in data used to train each classifier, they can be differentiated based on the generated models. The example of such interpretation is presented in Figure 7.1.

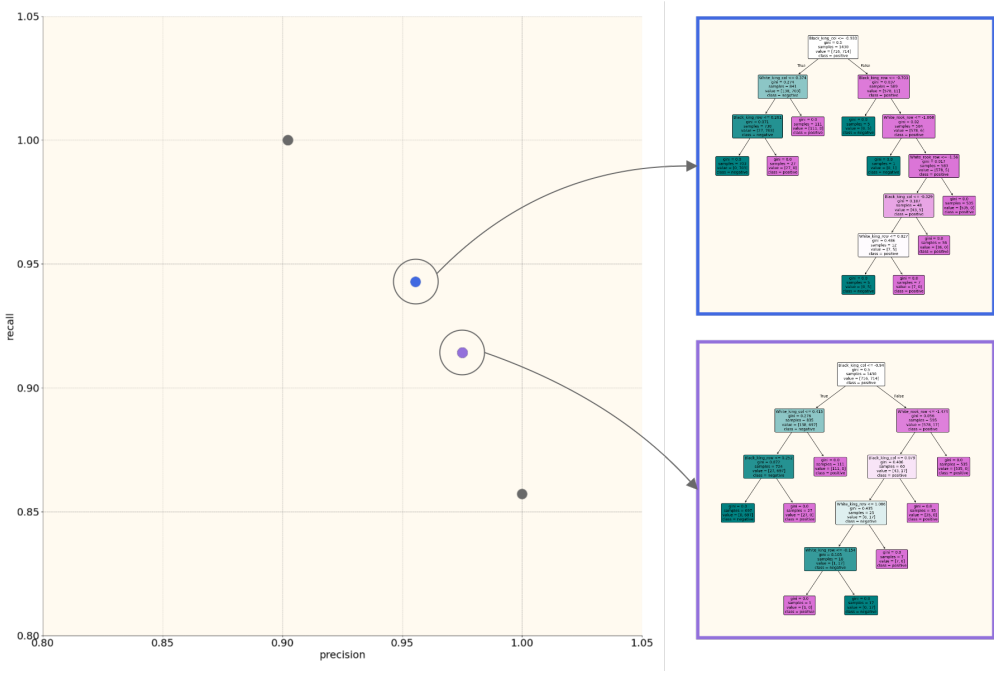


Figure 7.1: *Example of the tree solutions analysis*

Utilising interpretable models allows the user to understand the mechanism behind their prediction completely and to assess whether learned dependencies are accurate, by contrast to employing XAI, which may not explain the decisions correctly [188]. Moreover, it is pretty straightforward and does not require additional tools. However, there are also some significant disadvantages to this approach. Firstly, this technique is limited to the methods where the optimisation process results in (directly or indirectly) a single classifier. Furthermore, the prediction model needs to be interpretable, further restricting the algorithm selection. Otherwise, some explainability approach needs to be employed. Secondly, depending on the number of solutions, the analysis might be very complex and

time-consuming. Moreover, even when interpretable, models might be very complex, for example, in the case of the very deep decision trees.

7.3 Feature importance analysis

Another approach would be to analyse solutions from the Pareto front without dependence on the model's structure. It is possible to employ a model-agnostic XAI technique to explain the resulting classifier's decision. However, it is important to select an appropriate one. To simplify the analysis of the Pareto front, the chosen algorithm should offer the aggregated breakdown of the solutions, by, for example, determining the influence of each attribute on the final prediction. With this knowledge, the expert might eliminate the solutions where some features weigh more on the selected class, which might be caused by some bias or known false dependencies. It would also be beneficial in the context of fairness in a prediction model, as classifiers where sensitive attributes have too much influence might be discarded.

The method allowing for assessing the importance of each variable/feature was presented in [189] and is based on permutation testing. The version of the procedure conducted in this research is presented in Algorithm 3:

The premise of the method is that the more important the feature is, the bigger the impact it has on the prediction quality. Because of it, when the attribute's value does not match the real sample, the recognition of the objects degrades. To measure that decrease in quality, for each problem's feature the algorithm permutes the values between the samples and calculate the difference between chosen metric scores of model's prediction on original and permuted data. Usually, the final importance value equals either $L^0 - L^{*j}$ or $\frac{L^{*j}}{L^0}$. However, since the idea is to compare solutions from the Pareto front with different base quality, the terms were merged to show the magnitude of the feature's influence. Furthermore, since the algorithm involves randomness (by randomly permuting the values of one feature), the steps should be repeated multiple times [189]. In the case of the following analyses, the procedure was conducted 100 times for each feature, and the values were averaged.

Algorithm 3 Permutation based test to assess given feature importance

Input: Ψ - trained classifier

$X_{\mathcal{TS}}$ - test set matrix

$y_{\mathcal{TS}}$ - test set labels

l - loss function/quality measure

j - assessed feature index

k - number of comparison iterations

Output: vip_j - importance of the feature

$\hat{y}_{\mathcal{TS}} \leftarrow \Psi(X_{\mathcal{TS}})$

$l^0 \leftarrow l(y_{\mathcal{TS}}, \hat{y}_{\mathcal{TS}})$

$\overline{vip_j} \leftarrow \emptyset$

for $i \leftarrow 1$ to k **do**

$X^{*j}_{\mathcal{TS}} \leftarrow$ test set matrix with randomly perturbed j column

$\hat{y}^{*j}_{\mathcal{TS}} \leftarrow \Psi(X^{*j}_{\mathcal{TS}})$

$l^{*j} \leftarrow l(y_{\mathcal{TS}}, \hat{y}^{*j}_{\mathcal{TS}})$

$\overline{vip_j} \leftarrow \overline{vip_j} \cup \{\frac{l^0 - l^{*j}}{l^0}\}$

end for

$vip_j \leftarrow \frac{1}{\|\overline{vip_j}\|} \sum_{vip_{*j} \in \overline{vip_j}} vip_{*j}$

The benefit of employing only two criteria is that the solutions can be explicitly sorted in direct proportion to one objective (and in descending order to the other). Making advantage of that property, the plot presenting the influence of the feature based on the Pareto front placement can be proposed. The example of the plot, obtained by the COSMOS method with weighted cross-entropy on the dataset Bank Marketing [190], is presented in Figure 7.2.

The plot separately presents the change of each feature's importance based on the specific Pareto front solutions, where solutions are sorted in increasing order from the first objective. The importances are shown twofold. Firstly, the line of each plot indicates local changes of the attributes' influence on the final predictions, and it ranges between the smallest and largest value of the solutions' feature importance. It allows for determining trends in Pareto front solutions and limiting ranges based on selected variables. Secondly, the colours of the plot show the global importance, to assess which of the features influences the quality of prediction the most. In the example, it can be noticed that the importances of some attributes change correspondingly to the Pareto front solutions (*duration*, *pdays*, *cons.price.idx*, etc.), while only in some cases (mainly *duration* and *emp.var.rate*) this trend is significant to the final result.

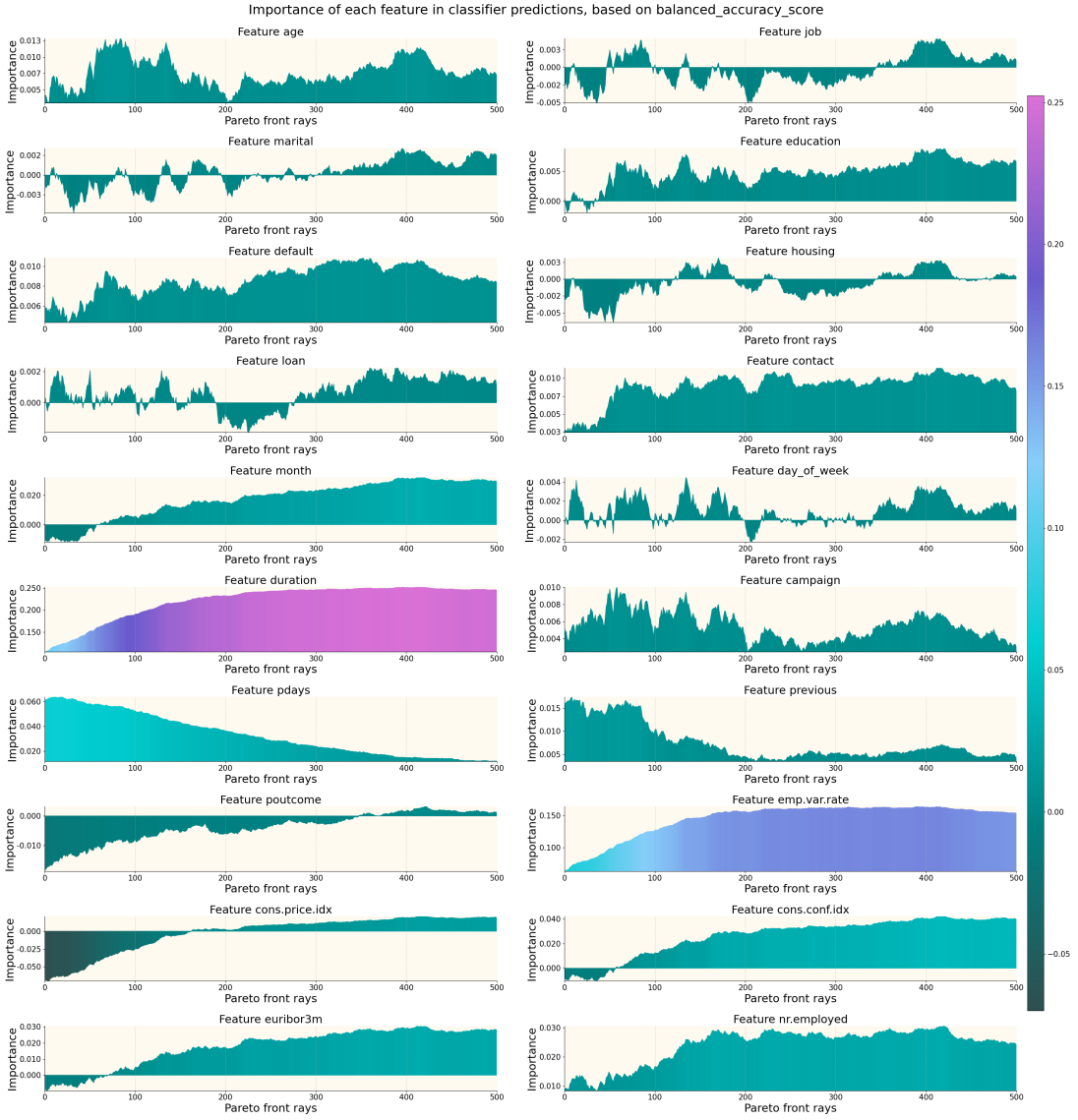


Figure 7.2: *Example of Pareto front features importance plot based on* BAC *measure*

The presented plot may be utilised in different ways:

1. ***Aiding solution selection*** - As mentioned before, a user with expert knowledge may take into consideration the importance of each feature while choosing the final model. For example, in the Figure 7.2, the feature with the most significant influence on the final prediction is *duration*. However, this variable accounts for the time of the phone call and is not known before the prediction (while being highly correlated to the label), so that a big importance might mean lower predictive abilities of the model. Furthermore, in the case of sensitive attributes, such as gender or race, the solutions with strong bias could also be eliminated.

2. ***Analysis of the model*** - In this thesis, MOO was usually employed to determine the set of parameters of a specific model/algorithm. Because of that, the feature importance plot allows analysis of the underlying method, since it shows the change in the behaviour based on different attributes. For example, in Figure 7.2, some solutions obtained negative importances in the case of features like *poutcome*, *cons.price.idx*. This might indicate that the features' influence is negative for the final prediction for solutions optimised more towards the second objective (in the example, minority class recognition). Moreover, importance might be assessed according to different quality metrics, meaning it could be analysed whether the algorithm is generally biased towards any of the classes (Figure 7.3).

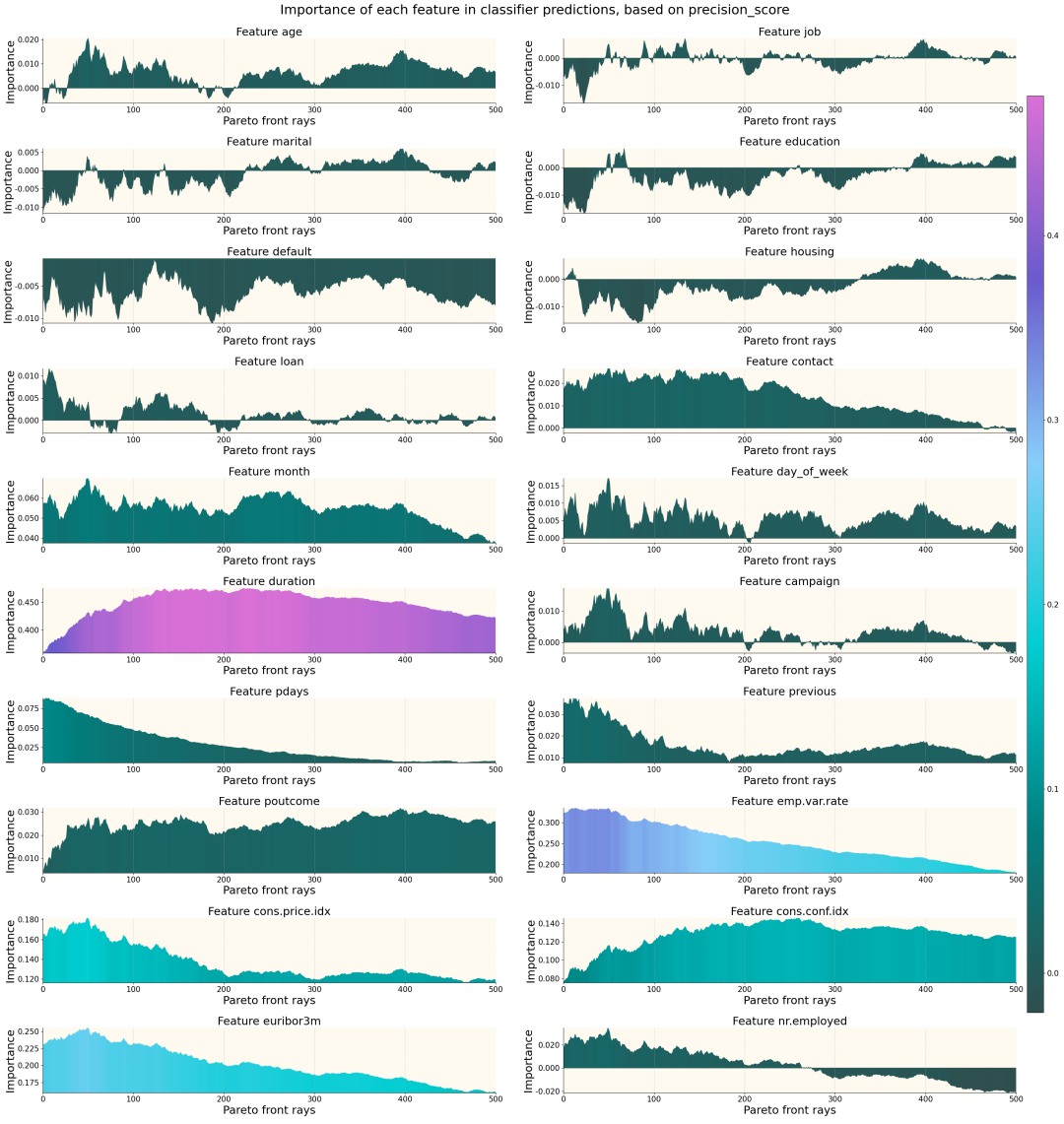


Figure 7.3: *Example of Pareto front features importance plot based on precision measure*

The downside of the proposed feature importance plot is its reliance on the base importance measure selection. As presented in Figures 7.2 and 7.3, the importances and even the trends of feature influence might vary significantly depending on what quality metric is chosen. Furthermore, the plot is mainly limited to the two objectives optimisation, as the visualisation of more criteria could be ambiguous and trends could also be hard to determine.

7.4 Lessons learnt

This chapter raised the subject of aiding the selection of the solution from the Pareto front, presenting concerns and difficulties with choice based only on objective values. The proposed approach was to employ XAI techniques that would allow deeper analysis and differentiation of the classifiers resulting from the optimisation. The most direct method was to explicitly utilise interpretable models, such as decision trees, to understand the basis of the prediction completely. However, this approach severely limited possible classification algorithms and required a long and complex breakdown of the solutions, as well as specific knowledge about the problem that might not be even feasible to obtain (such as particular dependencies of the problem's attributes). The alternative was the proposed feature importance plot, which presents the influence of the attribute on the final prediction quality, utilising a permutation test. The plot allows analysis of the solutions from the Pareto front, both in the context of model selection, where classifiers might be discarded when the feature weights are not appropriate, and the general algorithm behaviour breakdown. Its disadvantage is its reliance on two objectives and the selection of an importance measure.

Chapter 8

Conclusion and Future Works

This dissertation considered the application of multi-objective optimisation in creating methods dedicated to imbalanced data. Imbalanced data classification is a challenging and important task, which, if left unaddressed, has a significant impact on the correct recognition of rare samples. One of the difficulties of the problem is its assessment and optimisation goal, as usually utilised solutions based on a single criterion have their limitations on performance improvement. For this reason, the employment of multi-objective optimisation techniques was proposed, and the following hypothesis was formulated:

Incorporating MOO in training imbalanced data classifiers allows obtaining customised solutions whose quality is no worse than using SOO

To substantiate the hypothesis, several additional research questions were formulated and answered in consecutive dissertations' chapters.

Is it feasible to employ MOO in the process of training of the ensemble classifier, and how does it compare to the ensembles optimised using a single criterion?

In the chapter 3, the ensemble method utilising MOO was proposed. The committee consisted of several classification algorithms, which were trained on different subsets of the original data, obtained by stratified bootstrapping, to further ensure the diversity of the estimator pool. The NSGA II method was utilised to optimise the weights of classifiers being part of the ensemble, which are used during the final model's prediction via weighted majority voting. The selected objectives of the optimisation were *precision* and *recall*, so as to balance the influence of both classes' recognition quality. The conducted research proved the feasibility of employing MOO techniques with the proposed

criteria. However, the difficulties with a variety of Pareto front solutions for the smaller datasets were also exposed, arising from the limited possible values of utilised quality measures. The method was compared with ensembles created from the same estimator pool, where weights were optimised using SOO with different popular imbalanced data assessment metrics as objectives. The MOO-based models achieved results which were better or comparable to the ones obtained by ensembles based on aggregated measures, while also being more balanced in terms of *recall* and *precision*, in comparison to methods optimised specifically according to those metrics. The study confirmed the applicability of MOO in imbalanced data and its advantages over using a single criterion.

Is it possible to employ MOO in the preprocessing stage, and how does it improve the quality of a classification model

In the chapter 4, the sampling method utilising MOO was proposed. Data-based algorithms are very popular, since they are independent of a classifying model. Nevertheless, MOO is very rarely utilised in sampling approaches aside from feature and instance selection, even though it proved beneficial to the imbalanced data classification. The proposed algorithm determined minority class neighbourhoods from majority class samples and populated areas with synthetic minority class objects. The NSGA II algorithm was employed to optimise the sizes of the neighbourhoods (in the form of spheres) and the number of newly created samples. The method was compared to the most common baselines, as well as a classifier trained on original, not sampled data. The proposed algorithm outperformed most of the baselines and was marked by balanced results, achieving high scores for both classes recognition.

What is the best approach to estimate the quality criteria of the classifiers built using MOO?

In the chapter 5, the research was conducted to analyse different protocols of obtaining previously utilised optimisation objectives - *precision* and *recall*. Three methodologies were studied: hold-out, testing on training set and 5x2 cross-validation, with respect to the quality of estimated Pareto fronts, the estimation of the assessment on the test data and actual predictive abilities of the resulting models. The experiment showed that even though Pareto fronts generated by hold-out and testing on the training set dominated solutions by cross-validation, it was at the cost of smaller diversity and a bigger gap between training and validation scores. Furthermore, models generated by the latter had overall better prediction quality, although each of the protocols was superior for some datasets. In conclusion, cross-validation seems in general to be the best choice

of objective estimation, although for some tasks choosing a different approach could be beneficial or even necessary.

Is it possible to employ MOO gradient methods for the imbalanced data problem?

In the chapter 6, it was considered whether MOO gradient methods could be employed in place of the previously used genetic algorithm. The utilisation of weighted cross-entropy for objectives was proposed, with weights influencing the magnitude of influence of specific class recognition, while others had less of an impact. In results, two objectives were utilised - cross entropy minority and cross entropy majority, which aimed to substitute for respectively *recall* and *specificity*. Research was conducted to determine the best weights, the quality of Pareto fronts resulting from the proposed objective optimisation, their relation with target criteria and the quality of prediction of the method in comparison with a model optimised with a single loss function. The experiment proved the applicability of the proposed objectives and thus the MOO gradient approaches.

What is the diversity of classifiers from the Pareto front?

Finally, the chapter 7 considered the diversity of the Pareto front solutions and the approaches of their exploration. Previous research uncovered the importance of analysis of the classifiers resulting from multi-objective optimisation, so as to be able to consciously select an appropriate model. Two ways of exploring Pareto fronts were proposed - first based on utilising interpretable classification algorithms and second employing the XAI technique to determine a relationship between the problem's feature importance and placement of the solution. The examples showed possible trends in attribute importance variability, enabling both model analysis and solution selection based on their focus on specific variables.

To conclude, the conducted research resulted in answering the formulated research question and substantiating the research thesis.

Future work

During the preparation of this dissertation, a few new possible research directions were identified.

- Application of multi-objective optimisation in multi-class problems, which would also entail a necessity of developing new approaches of solution selection and analysis;
- Research on different MOO algorithms utilisation, for example, methods based on decomposition;
- Employment of different objectives suitable to imbalanced data, especially measures not based on prediction quality, to bypass classifier training and reduce computational complexity of the optimisation process;
- Research on new areas of MOO employment in imbalanced data classification tasks, for example, in feature selection or cost-sensitive learning;
- Application of MOO in new types of data, for example, imbalanced data streams;
- Incorporation of additional criteria, for example, related to diversity or problem constraints;
- Development of the method assisting decision making;
- Application of other techniques, i.e. XAI, in the imbalance data problem.

It can be noted that the last idea resulted in the project plan and receiving Grant no 2024/53/N/ST6/03667 (Preludium) funded by the Polish National Center.

Publications

The research presented in this thesis resulted in the following publications:

1. Węgier, W., Koziarski, M., Woźniak, M. (2022). Multicriteria classifier ensemble learning for imbalanced data. IEEE Access, 10, 16807-16818. (IF 3.9, MNiSW 100)
2. Węgier, W., Koziarski, M., Woźniak, M. (2023, July). Optimized hybrid imbalanced data sampling for decision tree training. In Proceedings of the Companion Conference on Genetic and Evolutionary Computation (pp. 339-342). (CORE A, MNiSW 140)

Additionally, research outside of the scope of this dissertation was also published:

1. Wegier, W., Ksieniewicz, P. (2020). Application of imbalanced data classification quality metrics as weighting methods of the ensemble data stream classification algorithms. Entropy, 22(8), 849. (IF 2.5, MNiSW 100)

2. Knapińska, A., Lechowicz, P., Węgier, W., Walkowiak, K. (2022). Long-term prediction of multiple types of time-varying network traffic using chunk-based ensemble learning. Applied Soft Computing, 130, 109694. (IF 8.7, MNiSW 200)

3. Wegier, W., Maczyński, M., Woźniak, M. (2024, December). Changing Lineup Classifier Ensemble for Drifting Imbalanced Data Streams. In 2024 IEEE International Conference on Data Mining Workshops (ICDMW) (pp. 238-245). IEEE. (CORE A*, MNiSW 200)

The research presented in this thesis was supported by the National Science Center, Poland under Grant 2019/35/B/ST6/04442.

Appendix

A Additional results for application of the multi-objective optimisation in ensemble learning

Table 8.1: *Precision results of compared ensembles*

	MOO					SOO							
DATASET	best precision	best recall	balanced	PROMETHEE precision	PROMETHEE recall	BAC	precision	recall	f1 score	auc	Gmean	AdaBoost	Bagging
abalone9-18	**0.811**	0.811	0.811	0.811	0.811	0.672	0.763	0.538	0.661	0.691	0.59	0.485	0.586
aps failure	0.850	0.448	0.738	0.850	0.448	0.469	**0.882**	0.208	0.740	0.506	0.436	0.764	0.867
covid	**0.925**	0.419	0.425	0.925	0.419	0.129	0.826	0.021	0.478	0.130	0.121	0.624	0.674
credit card	0.951	0.900	0.900	0.951	0.900	0.887	**0.970**	0.434	0.894	0.884	0.907	0.823	0.914
diabetes	0.426	0.147	0.185	0.426	0.147	0.140	0.269	0.112	0.143	0.167	0.142	**0.472**	0.407
ecoli1	0.765	0.706	0.761	0.765	0.706	0.738	0.731	0.674	0.728	0.715	0.741	0.759	**0.793**
ecoli2	0.855	0.852	0.852	0.855	0.852	0.846	0.869	0.795	0.836	**0.880**	0.838	0.805	0.840
ecoli3	0.693	0.695	0.695	0.693	0.695	0.662	0.646	0.601	0.665	0.653	0.645	0.511	**0.711**
flare-F	0.271	0.327	0.331	0.271	0.327	0.310	0.318	0.132	0.307	0.271	0.272	**0.409**	0.172
glass0	0.700	0.692	0.690	0.700	0.692	0.676	0.687	0.608	0.665	0.685	0.690	0.730	**0.779**
glass1	0.732	0.727	0.727	0.732	0.727	0.696	0.704	0.564	0.693	0.692	0.695	0.669	**0.768**
glass4	0.499	0.499	0.499	0.499	0.499	0.440	0.525	0.473	0.352	0.257	0.467	**0.734**	0.695
glass5	0.088	0.088	0.088	0.088	0.088	0.000	0.000	0.000	0.000	0.000	0.000	**0.69**	0.621
haberman	0.514	0.537	0.537	0.514	0.537	**0.550**	0.540	0.491	0.515	0.543	0.512	0.391	0.391
hand positions	0.990	0.900	0.912	0.990	0.900	0.934	**0.991**	0.874	0.963	0.938	0.932	0.805	0.970
miniboone	0.917	0.362	0.875	0.917	0.362	0.806	**0.982**	0.282	0.854	0.807	0.808	0.847	0.886
mitbih	**0.964**	0.948	0.948	0.964	0.948	0.738	0.962	0.576	0.946	0.827	0.783	0.865	0.953
page-blocks	0.900	0.887	0.887	0.900	0.887	0.888	**0.906**	0.806	0.883	0.877	0.889	0.834	0.882
pima	0.678	0.664	0.664	0.678	0.664	0.691	**0.701**	0.667	0.684	0.685	0.678	0.647	0.656
vehicle1	0.594	0.570	0.578	0.594	0.570	0.584	0.604	0.438	0.570	0.590	0.574	0.568	**0.618**
vehicle3	0.604	0.554	0.576	0.604	0.554	0.559	**0.614**	0.436	0.561	0.552	0.538	0.594	0.604
yeast1	0.647	0.366	0.569	0.647	0.366	0.559	**0.654**	0.299	0.572	0.544	0.565	0.612	0.621
yeast3	0.787	0.693	0.772	0.787	0.693	0.758	**0.797**	0.216	0.767	0.754	0.716	0.751	0.777
yeast4	0.527	0.358	0.394	0.527	0.358	0.464	0.475	0.138	0.456	0.416	0.441	0.332	**0.530**
yeast5	0.711	0.625	0.668	0.711	0.625	0.633	0.737	0.531	0.664	0.670	0.640	0.711	**0.757**

Table 8.2: *Recall results of compared ensembles*

	MOO					SOO							
DATASET	best precision	best recall	balanced	PROMETHEE precision	PROMETHEE recall	BAC	precision	recall	f1 score	auc	Gmean	AdaBoost	Bagging
abalone9-18	0.176	0.176	0.176	0.176	0.176	0.081	0.138	0.205	0.152	0.133	0.152	**0.357**	0.229
aps failure	0.581	0.868	0.736	0.581	0.868	0.860	0.496	**0.918**	0.725	0.848	0.866	0.647	0.695
covid	0.060	0.326	0.325	0.060	0.326	0.778	0.007	**0.999**	0.416	0.804	0.748	0.234	0.296
credit card	0.587	0.746	0.746	0.587	0.746	0.740	0.304	**0.854**	0.745	0.739	0.716	0.682	0.775
diabetes	0.002	0.345	0.174	0.002	0.345	0.219	0.001	**1.000**	0.436	0.290	0.549	0.013	0.04
ecoli1	0.738	0.805	0.769	0.738	0.805	0.788	0.777	**0.832**	0.803	0.800	0.767	0.749	0.741
ecoli2	0.815	0.823	0.823	0.815	0.823	0.823	0.812	**0.854**	0.812	0.800	0.815	0.735	0.735
ecoli3	0.544	0.549	0.549	0.544	0.549	0.548	0.567	**0.636**	0.590	0.533	0.543	0.522	0.557
flare-F	0.079	0.251	0.242	0.079	0.251	0.154	0.093	**0.702**	0.201	0.173	0.213	0.164	0.126
glass0	0.743	0.797	0.774	0.743	0.797	0.809	0.771	**0.840**	0.800	0.789	0.794	0.686	0.746
glass1	0.542	0.610	0.610	0.542	0.610	0.563	0.460	**0.666**	0.547	0.566	0.582	0.637	0.655
glass4	0.202	0.202	0.202	0.202	0.202	0.126	0.112	0.19	0.148	0.098	0.205	**0.624**	0.448
glass5	0.125	0.125	0.125	0.125	0.125	0.000	0.000	0.000	0.000	0.000	0.000	**0.620**	0.420
haberman	0.254	0.304	0.304	0.254	0.304	0.272	0.26	**0.329**	0.262	0.274	0.257	0.267	0.294
hand positions	0.783	0.852	0.851	0.783	0.852	0.851	0.778	0.858	0.839	0.848	0.849	0.589	**0.880**
miniboone	0.712	0.999	0.874	0.712	0.999	0.940	0.156	**1.000**	0.898	0.937	0.936	0.837	0.857
mitbih	0.589	0.600	0.600	0.589	0.600	0.643	0.591	**0.691**	0.600	0.628	0.637	0.431	0.582
page-blocks	0.777	0.800	0.800	0.777	0.800	0.789	0.738	0.821	0.797	0.798	0.790	0.790	**0.842**
pima	0.552	0.558	0.558	0.552	0.558	0.548	0.540	0.569	0.558	0.554	0.558	**0.590**	0.565
vehicle1	0.423	0.504	0.497	0.423	0.504	0.491	0.414	**0.637**	0.472	0.480	0.505	0.432	0.431
vehicle3	0.400	0.516	0.488	0.400	0.516	0.465	0.386	**0.611**	0.473	0.497	0.499	0.425	0.380
yeast1	0.435	0.949	0.604	0.435	0.949	0.627	0.418	**0.987**	0.586	0.638	0.622	0.466	0.454
yeast3	0.763	0.836	0.792	0.763	0.836	0.799	0.761	**0.961**	0.778	0.810	0.835	0.730	0.697
yeast4	0.201	0.423	0.368	0.201	0.423	0.298	0.184	**0.679**	0.309	0.361	0.274	0.203	0.224
yeast5	0.586	0.668	0.65	0.586	0.668	0.691	0.523	**0.755**	0.636	0.659	0.673	0.682	0.550

Table 8.3: BAC *results of compared ensembles*

	MOO					SOO							
DATASET	best precision	best recall	balanced	PROMETHEE precision	PROMETHEE recall	BAC	precision	recall	f1 score	auc	Gmean	AdaBoost	Bagging
abalone9-18	0.585	0.585	0.585	0.585	0.585	0.540	0.566	0.592	0.571	0.563	0.572	**0.666**	0.609
aps failure	0.790	**0.924**	0.866	0.790	0.924	0.921	0.747	0.770	0.860	0.916	0.922	0.822	0.846
covid	0.530	0.658	0.657	0.530	0.658	0.831	0.504	0.500	0.703	**0.844**	0.815	0.615	0.647
credit card	0.793	0.873	0.873	0.793	0.873	0.870	0.652	0.748	0.872	0.869	0.858	0.841	**0.888**
diabetes	0.501	0.519	0.539	0.501	0.519	0.538	0.500	0.500	0.548	0.549	**0.565**	0.505	0.516
ecoli1	0.834	0.852	0.848	0.834	0.852	0.851	0.845	0.854	**0.856**	0.852	0.842	0.838	0.840
ecoli2	0.894	0.897	0.897	0.894	0.897	0.897	0.894	**0.906**	0.891	0.889	0.893	0.850	0.854
ecoli3	0.757	0.760	0.760	0.757	0.760	0.757	0.766	**0.792**	0.777	0.749	0.753	0.732	0.765
flare-F	0.534	0.612	0.608	0.534	0.612	0.566	0.541	**0.725**	0.588	0.576	0.591	0.577	0.552
glass0	0.791	0.810	0.800	0.791	0.810	0.809	0.798	0.784	0.799	0.803	0.808	0.779	**0.819**
glass1	0.716	0.740	0.740	0.716	0.740	0.711	0.675	0.674	0.700	0.713	0.716	0.733	**0.771**
glass4	0.597	0.597	0.597	0.597	0.597	0.561	0.554	0.592	0.570	0.544	0.599	**0.802**	0.714
glass5	0.560	0.560	0.560	0.560	0.560	0.500	0.500	0.500	0.500	0.500	0.500	**0.803**	0.705
haberman	0.585	**0.603**	0.603	0.585	0.603	0.594	0.588	0.599	0.586	0.594	0.583	0.559	0.566
hand positions	0.891	0.915	0.916	0.891	0.915	0.919	0.888	0.914	0.916	0.917	0.917	0.778	**0.937**
miniboone	0.842	0.646	0.913	0.842	0.646	**0.925**	0.578	0.503	0.919	0.925	0.924	0.889	0.907
mitbih	0.794	0.800	0.800	0.794	0.800	**0.816**	0.795	0.666	0.800	0.810	0.814	0.715	0.791
page-blocks	0.883	0.894	0.894	0.883	0.894	0.889	0.864	0.895	0.892	0.893	0.889	0.886	**0.915**
pima	0.705	0.703	0.703	0.705	0.703	0.707	0.707	0.708	**0.709**	0.708	0.708	0.708	0.702
vehicle1	0.661	0.686	0.686	0.661	0.686	0.685	0.660	0.659	0.674	0.682	**0.686**	0.659	0.668
vehicle3	0.655	**0.688**	0.683	0.655	0.688	0.670	0.652	0.651	0.674	0.680	0.677	0.664	0.648
yeast1	0.668	0.639	0.708	0.668	0.639	0.709	0.663	0.523	0.701	0.708	**0.712**	0.673	0.670
yeast3	0.869	0.892	0.881	0.869	0.892	0.883	0.868	0.676	0.874	0.888	**0.895**	0.850	0.836
yeast4	0.596	**0.691**	0.672	0.596	0.691	0.642	0.589	0.690	0.648	0.671	0.629	0.594	0.607
yeast5	0.789	0.827	0.820	0.789	0.827	0.838	0.758	**0.857**	0.813	0.824	0.830	0.836	0.772

Table 8.4: *F1 score results of compared ensembles*

	MOO					SOO							
DATASET	best precision	best recall	balanced	PROMETHEE precision	PROMETHEE recall	BAC	precision	recall	f1 score	auc	Gmean	AdaBoost	Bagging
abalone9-18	0.249	0.249	0.249	0.249	0.249	0.138	0.202	0.236	0.208	0.193	0.194	**0.399**	0.316
aps failure	0.687	0.587	0.737	0.687	0.587	0.606	0.628	0.307	0.732	0.631	0.577	0.700	**0.771**
covid	0.111	0.362	0.365	0.111	0.362	0.221	0.014	0.041	**0.443**	0.223	0.208	0.340	0.411
credit card	0.720	0.815	0.815	0.720	0.815	0.804	0.430	0.424	0.811	0.802	0.799	0.745	**0.839**
diabetes	0.003	0.169	0.178	0.003	0.169	0.149	0.002	0.201	0.211	0.192	**0.225**	0.025	0.072
ecoli1	0.747	0.750	**0.762**	0.747	0.750	0.758	0.751	0.741	0.762	0.752	0.750	0.750	0.762
ecoli2	0.830	0.832	0.832	0.830	0.832	0.830	**0.835**	0.821	0.821	0.833	0.824	0.762	0.781
ecoli3	0.600	0.604	0.604	0.600	0.604	0.590	0.594	0.598	**0.621**	0.572	0.580	0.511	0.619
flare-F	0.112	0.243	**0.244**	0.112	0.243	0.161	0.131	0.210	0.219	0.192	0.198	0.226	0.141
glass0	0.710	0.738	0.723	0.710	0.738	0.735	0.722	0.702	0.723	0.729	0.736	0.703	**0.757**
glass1	0.619	0.660	0.660	0.619	0.660	0.616	0.548	0.597	0.603	0.619	0.625	0.650	**0.704**
glass4	0.271	0.271	0.271	0.271	0.271	0.187	0.179	0.250	0.196	0.132	0.264	**0.642**	0.509
glass5	0.102	0.102	0.102	0.102	0.102	0.000	0.000	0.000	0.000	0.000	0.000	**0.583**	0.442
haberman	0.337	**0.383**	0.383	0.337	0.383	0.358	0.344	0.381	0.343	0.358	0.334	0.315	0.332
hand positions	0.874	0.874	0.880	0.874	0.874	0.891	0.872	0.865	0.897	0.891	0.888	0.680	**0.923**
miniboone	0.786	0.529	0.875	0.786	0.529	0.867	0.260	0.440	**0.875**	0.867	0.867	0.842	0.871
mitbih	0.731	**0.735**	0.735	0.731	0.735	0.646	0.732	0.458	0.734	0.687	0.668	0.575	0.723
page-blocks	0.833	0.840	0.840	0.833	0.840	0.835	0.813	0.801	0.837	0.834	0.836	0.810	**0.862**
pima	0.607	0.604	0.604	0.607	0.604	0.609	0.608	0.613	0.613	0.611	0.611	**0.616**	0.606
vehicle1	0.490	0.530	**0.531**	0.490	0.530	0.528	0.486	0.506	0.510	0.521	0.530	0.489	0.501
vehicle3	0.476	**0.532**	0.527	0.476	0.532	0.503	0.472	0.498	0.511	0.519	0.515	0.494	0.465
yeast1	0.515	0.528	0.584	0.515	0.528	0.585	0.506	0.459	0.575	0.585	**0.590**	0.528	0.523
yeast3	0.773	0.748	0.779	0.773	0.748	0.776	0.777	0.327	0.771	**0.779**	0.765	0.739	0.734
yeast4	0.255	0.331	0.353	0.255	0.331	0.341	0.249	0.193	0.359	**0.370**	0.317	0.248	0.289
yeast5	0.633	0.623	0.646	0.633	0.623	0.641	0.588	0.576	0.630	0.658	0.646	**0.684**	0.626

Table 8.5: AUC *results of compared ensembles*

	MOO					SOO							
DATASET	best precision	best recall	balanced	PROMETHEE precision	PROMETHEE recall	BAC	precision	recall	f1 score	auc	Gmean	AdaBoost	Bagging
abalone9-18	0.585	0.585	0.585	0.585	0.585	0.540	0.566	0.592	0.571	0.563	0.572	**0.666**	0.609
aps failure	0.790	**0.924**	0.866	0.790	0.924	0.921	0.747	0.770	0.860	0.916	0.922	0.822	0.846
covid	0.530	0.658	0.657	0.530	0.658	0.831	0.504	0.500	0.703	**0.844**	0.815	0.615	0.647
credit card	0.793	0.873	0.873	0.793	0.873	0.870	0.652	0.748	0.872	0.869	0.858	0.841	**0.888**
diabetes	0.501	0.519	0.539	0.501	0.519	0.538	0.500	0.500	0.548	0.549	**0.565**	0.505	0.516
ecoli1	0.834	0.852	0.848	0.834	0.852	0.851	0.845	0.854	**0.856**	0.852	0.842	0.838	0.840
ecoli2	0.894	0.897	0.897	0.894	0.897	0.897	0.894	**0.906**	0.891	0.889	0.893	0.850	0.854
ecoli3	0.757	0.760	0.760	0.757	0.760	0.757	0.766	**0.792**	0.777	0.749	0.753	0.732	0.765
flare-F	0.534	0.612	0.608	0.534	0.612	0.566	0.541	**0.725**	0.588	0.576	0.591	0.577	0.552
glass0	0.791	0.810	0.800	0.791	0.810	0.809	0.798	0.784	0.799	0.803	0.808	0.779	**0.819**
glass1	0.716	0.740	0.740	0.716	0.740	0.711	0.675	0.674	0.700	0.713	0.716	0.733	**0.771**
glass4	0.597	0.597	0.597	0.597	0.597	0.561	0.554	0.592	0.57	0.544	0.599	**0.802**	0.714
glass5	0.560	0.560	0.560	0.560	0.560	0.500	0.500	0.500	0.500	0.500	0.500	**0.803**	0.705
haberman	0.585	**0.603**	0.603	0.585	0.603	0.594	0.588	0.599	0.586	0.594	0.583	0.559	0.566
hand positions	0.891	0.915	0.916	0.891	0.915	0.919	0.888	0.914	0.916	0.917	0.917	0.778	**0.937**
miniboone	0.842	0.646	0.913	0.842	0.646	**0.925**	0.578	0.503	0.919	0.925	0.924	0.889	0.907
mitbih	0.794	0.800	0.800	0.794	0.800	**0.816**	0.795	0.666	0.8	0.81	0.814	0.715	0.791
page-blocks	0.883	0.894	0.894	0.883	0.894	0.889	0.864	0.895	0.892	0.893	0.889	0.886	**0.915**
pima	0.705	0.703	0.703	0.705	0.703	0.707	0.707	0.708	**0.709**	0.708	0.708	0.708	0.702
vehicle1	0.661	0.686	0.686	0.661	0.686	0.685	0.66	0.659	0.674	0.682	**0.686**	0.659	0.668
vehicle3	0.655	**0.688**	0.683	0.655	0.688	0.670	0.652	0.651	0.674	0.680	0.677	0.664	0.648
yeast1	0.668	0.639	0.708	0.668	0.639	0.709	0.663	0.523	0.701	0.708	**0.712**	0.673	0.670
yeast3	0.869	0.892	0.881	0.869	0.892	0.883	0.868	0.676	0.874	0.888	**0.895**	0.850	0.836
yeast4	0.596	**0.691**	0.672	0.596	0.691	0.642	0.589	0.69	0.648	0.671	0.629	0.594	0.607
yeast5	0.789	0.827	0.82	0.789	0.827	0.838	0.758	**0.857**	0.813	0.824	0.83	0.836	0.772

Table 8.6: *Gmean results of compared ensembles*

	MOO					SOO							
DATASET	best precision	best recall	balanced	PROMETHEE precision	PROMETHEE recall	BAC	precision	recall	f1 score	auc	Gmean	AdaBoost	Bagging
abalone9-18	0.393	0.393	0.393	0.393	0.393	0.243	0.341	0.396	0.345	0.337	0.311	**0.583**	0.466
aps failure	0.687	0.587	0.737	0.687	0.587	0.606	0.628	0.307	0.732	0.631	0.577	0.803	**0.833**
covid	0.241	0.568	0.567	0.241	0.568	**0.829**	0.075	0.004	0.641	0.843	0.811	0.483	0.543
credit card	0.763	0.863	0.863	0.763	0.863	0.859	0.520	0.596	0.863	0.859	0.845	0.825	**0.880**
diabetes	0.038	0.343	0.395	0.038	0.343	0.350	0.018	0.002	0.521	0.455	**0.564**	0.112	0.198
ecoli1	0.827	0.850	0.843	0.827	0.850	0.847	0.841	0.853	**0.854**	0.849	0.838	0.832	0.833
ecoli2	0.889	0.893	0.893	0.889	0.893	0.892	0.889	**0.904**	0.886	0.883	0.888	0.840	0.845
ecoli3	0.722	0.726	0.726	0.722	0.726	0.723	0.732	**0.767**	0.751	0.711	0.718	0.696	0.733
flare-F	0.240	0.450	0.445	0.240	0.450	0.325	0.277	**0.710**	0.423	0.389	0.424	0.393	0.321
glass0	0.783	**0.809**	0.796	**0.783**	0.809	0.808	0.795	0.780	0.798	0.801	0.807	0.772	0.813
glass1	0.691	0.727	0.727	0.691	0.727	0.692	0.633	0.661	0.679	0.695	0.699	0.724	**0.761**
glass4	0.363	0.363	0.363	0.363	0.363	0.247	0.255	0.323	0.263	0.193	0.342	**0.771**	0.649
glass5	0.148	0.148	0.148	0.148	0.148	0.000	0.000	0.000	0.000	0.000	0.000	**0.716**	0.578
haberman	0.477	**0.521**	**0.521**	0.477	**0.521**	0.495	0.483	0.526	0.485	0.496	0.476	0.473	0.489
hand positions	0.884	0.912	0.914	0.884	0.912	**0.916**	0.881	0.913	0.912	0.915	0.915	0.754	0.935
miniboone	0.824	0.522	0.912	0.824	0.522	**0.925**	0.381	0.081	0.918	**0.925**	0.924	0.887	0.905
mitbih	0.767	0.775	0.775	0.767	0.775	**0.796**	0.769	0.575	0.774	0.789	0.794	0.656	0.763
page-blocks	0.877	0.889	0.889	0.877	0.889	0.883	0.855	0.891	0.887	0.887	0.884	0.880	**0.912**
pima	0.687	0.686	0.686	0.687	0.686	0.688	0.686	0.693	0.692	0.690	0.691	**0.697**	0.688
vehicle1	0.613	**0.658**	0.656	0.613	0.658	0.652	0.608	0.643	0.638	0.645	**0.658**	0.617	0.621
vehicle3	0.600	**0.664**	0.654	0.600	**0.664**	0.635	0.593	0.631	0.642	0.653	0.651	0.618	0.589
yeast1	0.623	0.555	0.699	0.623	0.555	0.701	0.613	0.235	0.690	0.703	**0.705**	0.640	0.634
yeast3	0.862	0.889	0.876	0.862	0.889	0.879	0.861	0.584	0.868	0.884	**0.892**	0.840	0.824
yeast4	0.425	0.614	0.587	0.425	0.614	0.532	0.390	**0.662**	0.546	0.586	0.510	0.438	0.449
yeast5	0.758	0.805	0.797	0.758	0.805	0.821	0.709	**0.845**	0.787	0.805	0.811	0.818	0.734
yeast6	0.557	0.613	0.577	0.557	0.613	0.582	0.561	**0.735**	0.592	0.528	0.574	0.617	0.575

B Additional results for application of multi-objective optimisation in data sampling

Table 8.7: *Precision results of compared algorithms for* CART *model.*

DATASET	Bare classifier	ROS	SMOTE	Borderline SMOTE	ADASYN	CCR	balanced	best precision	best recall
page-blocks-1-3 vs 4	0.885	**0.933**	0.915	0.774	0.925	0.826	0.842	0.864	0.845
yeast-0-5-6-7-9 vs 4	0.360	0.370	0.320	0.359	0.35	0.362	0.363	**0.399**	0.373
yeast-1-2-8-9 vs 7	0.204	0.182	0.094	0.177	0.096	**0.221**	0.167	0.167	0.176
yeast-1-4-5-8 vs 7	0.114	**0.116**	0.077	0.099	0.087	0.104	0.092	0.107	0.113
yeast-1 vs 7	0.318	0.294	0.180	0.225	0.185	0.294	0.314	**0.322**	0.293
yeast-2 vs 4	0.683	0.711	0.683	0.618	0.651	**0.720**	0.668	0.716	0.678
yeast-2 vs 8	0.507	0.533	0.418	0.476	0.343	**0.536**	0.438	0.453	0.395
yeast4	0.260	0.290	0.214	0.209	0.208	**0.298**	0.252	0.227	0.269
yeast5	0.642	**0.673**	0.658	0.531	0.666	0.601	0.596	0.589	0.609
yeast6	0.367	**0.454**	0.283	0.341	0.286	0.361	0.398	0.383	0.330
ecoli-0-1-4-7 vs 2-3-5-6	0.583	**0.656**	0.488	0.596	0.520	0.592	0.603	0.598	0.600
ecoli-0-1 vs 2-3-5	0.727	0.723	0.624	**0.741**	0.611	0.593	0.689	0.68	0.713
ecoli-0-2-6-7 vs 3-5	0.629	**0.665**	0.586	0.557	0.579	0.634	0.653	0.642	0.630
ecoli-0-6-7 vs 3-5	**0.708**	0.668	0.569	0.583	0.541	0.697	0.631	0.599	0.691
ecoli-0-6-7 vs 5	**0.791**	0.770	0.686	0.656	0.589	0.781	0.755	0.747	0.755
yeast-0-2-5-6 vs 3-7-8-9	0.465	**0.509**	0.385	0.402	0.356	0.445	0.442	0.439	0.454
yeast-0-3-5-9 vs 7-8	**0.343**	0.320	0.224	0.248	0.204	0.328	0.306	0.292	0.298
abalone-17 vs 7-8-9-10	**0.271**	0.263	0.161	0.196	0.180	0.180	0.228	0.259	0.158
abalone-19 vs 10-11-12-13	0.074	**0.082**	0.055	0.044	0.052	0.066	0.073	0.075	0.060
abalone-20 vs 8-9-10	**0.229**	0.261	0.168	0.228	0.166	0.173	0.191	0.202	0.136
flare-F	**0.233**	0.140	0.196	0.156	0.179	0.151	0.228	0.226	0.219
kr-vs-k-zero vs eight	**0.910**	0.871	0.907	0.843	0.877	0.894	0.859	0.836	0.878
poker-8-9 vs 5	0.086	0.104	0.071	0.074	0.066	0.097	**0.105**	0.068	0.080
poker-8-9 vs 6	0.190	0.109	0.285	0.170	0.262	0.469	**0.827**	0.746	0.741
poker-8 vs 6	0.309	0.389	0.513	0.297	0.502	0.476	0.667	**0.746**	0.697
winequality-red-4	0.105	0.091	0.101	0.103	0.065	0.101	0.092	**0.113**	0.102
winequality-white-3-9 vs 5	**0.197**	0.073	0.044	0.164	0.036	0.073	0.158	0.131	0.118
winequality-white-3 vs 7	0.304	0.164	0.068	**0.314**	0.056	0.105	0.281	0.253	0.169
ecoli1	0.708	0.721	0.707	0.700	0.713	0.724	0.715	**0.731**	0.692
ecoli2	0.718	0.747	0.723	0.713	0.674	**0.757**	0.716	0.740	0.719
ecoli3	**0.591**	0.448	0.459	0.517	0.507	0.502	0.547	0.575	0.493
glass0	0.628	**0.691**	0.665	0.642	0.651	0.667	0.667	**0.691**	0.628
glass1	**0.649**	0.641	0.625	0.634	0.581	0.637	0.630	0.639	0.611
haberman	0.357	0.343	0.346	0.364	0.357	**0.399**	0.375	0.353	0.337
pima	0.573	0.559	0.563	0.543	0.538	**0.575**	0.546	0.541	0.563
yeast3	0.697	0.697	0.660	0.656	0.658	**0.718**	0.679	0.687	0.687

Table 8.8: *Recall results of compared algorithms for* CART *model.*

DATASET	Bare classifier	ROS	SMOTE	Borderline SMOTE	ADASYN	CCR	balanced	best precision	best recall
page-blocks-1-3 vs 4	0.871	**0.979**	**0.979**	0.821	0.964	0.957	0.936	0.957	0.943
yeast-0-5-6-7-9 vs 4	0.401	0.400	0.489	0.478	**0.538**	0.395	0.436	0.448	0.451
yeast-1-2-8-9 vs 7	0.247	0.180	0.220	0.247	**0.233**	0.273	0.247	0.247	0.267
yeast-1-4-5-8 vs 7	0.14	0.107	0.173	0.127	**0.193**	0.133	0.133	0.140	0.173
yeast-1 vs 7	0.373	0.273	0.300	0.307	0.313	0.340	**0.42**	0.413	0.380
yeast-2 vs 4	0.691	0.663	**0.757**	0.683	0.749	0.753	0.71	0.698	0.705
yeast-2 vs 8	0.500	0.480	**0.610**	0.500	0.550	0.560	0.500	0.510	0.490
yeast4	0.286	0.284	0.393	0.302	0.382	0.329	0.373	0.326	**0.404**
yeast5	0.614	0.632	0.732	0.641	**0.750**	0.691	0.691	0.705	**0.750**
yeast6	0.436	0.420	0.505	0.476	0.521	0.522	0.515	0.526	**0.573**
ecoli-0-1-4-7 vs 2-3-5-6	0.608	0.578	0.668	0.635	0.660	0.641	0.660	**0.695**	0.676
ecoli-0-1 vs 2-3-5	0.550	0.583	0.608	0.575	0.650	0.642	0.692	**0.708**	**0.708**
ecoli-0-2-6-7 vs 3-5	0.627	0.573	0.655	0.527	0.664	0.655	0.664	**0.700**	0.636
ecoli-0-6-7 vs 3-5	0.664	0.627	0.682	0.700	0.709	0.664	0.691	**0.718**	0.664
ecoli-0-6-7 vs 5	0.670	0.620	0.700	0.580	0.590	0.750	0.720	0.750	**0.770**
yeast-0-2-5-6 vs 3-7-8-9	0.496	0.465	0.505	0.481	0.501	0.491	0.514	0.511	**0.554**
yeast-0-3-5-9 vs 7-8	0.336	0.308	**0.368**	0.332	0.316	0.360	0.344	0.340	0.344
abalone-17 vs 7-8-9-10	0.317	0.245	0.352	0.279	0.400	0.328	0.290	0.314	**0.479**
abalone-19 vs 10-11-12-13	0.100	0.062	0.206	0.069	0.194	0.081	0.138	0.144	**0.294**
abalone-20 vs 8-9-10	0.269	0.231	**0.423**	0.285	**0.423**	0.331	0.285	0.308	0.362
flare-F	0.163	0.261	0.201	0.242	0.182	**0.303**	0.218	0.205	0.209
kr-vs-k-zero vs eight	0.954	0.830	0.946	0.813	0.878	0.909	**0.962**	0.910	**0.962**
poker-8-9 vs 5	0.105	0.096	0.185	0.088	0.158	0.099	0.134	0.079	**0.198**
poker-8-9 vs 6	0.174	0.079	0.315	0.209	0.338	0.432	**0.855**	0.849	0.801
poker-8 vs 6	0.210	0.210	0.389	0.229	0.417	0.404	**0.631**	0.629	0.629
winequality-red-4	0.127	0.097	0.257	0.135	0.166	0.211	0.139	0.163	**0.288**
winequality-white-3-9 vs 5	0.169	0.073	0.137	0.178	0.119	0.137	**0.194**	0.178	**0.194**
winequality-white-3 vs 7	0.220	0.120	0.150	0.240	0.120	0.160	**0.260**	0.220	0.170
ecoli1	0.723	0.728	0.760	0.778	0.780	0.744	0.77	0.775	**0.793**
ecoli2	0.712	0.750	0.758	0.704	0.769	0.769	**0.781**	0.758	0.762
ecoli3	0.556	0.453	0.589	0.594	0.624	0.584	0.657	0.675	**0.713**
glass0	0.686	0.711	0.726	0.731	0.726	0.711	**0.783**	0.774	0.780
glass1	0.661	0.655	0.666	0.692	0.616	0.676	0.687	0.700	**0.708**
haberman	0.368	0.331	0.422	0.410	0.412	0.484	0.506	0.481	**0.587**
pima	0.58	0.584	0.604	0.598	0.565	**0.613**	0.578	0.589	0.581
yeast3	0.692	0.687	0.747	0.710	0.765	0.726	0.756	0.756	**0.774**

Table 8.9: BAC *results of compared algorithms for* CART *model.*

DATASET	Bare classifier	ROS	SMOTE	Borderline SMOTE	ADASYN	CCR	balanced	best precision	best recall
page-blocks-1-3 vs 4	0.932	**0.987**	0.986	0.903	0.980	0.971	0.962	0.973	0.965
yeast-0-5-6-7-9 vs 4	0.662	0.663	0.689	0.693	**0.715**	0.660	0.677	0.687	0.685
yeast-1-2-8-9 vs 7	0.607	0.576	0.573	0.603	0.582	**0.620**	0.604	0.604	0.613
yeast-1-4-5-8 vs 7	0.545	0.534	0.538	0.535	0.548	0.540	0.536	0.544	**0.557**
yeast-1 vs 7	0.659	0.613	0.600	0.616	0.607	0.641	**0.677**	0.674	0.655
yeast-2 vs 4	0.827	0.816	0.859	0.817	0.852	**0.860**	0.835	0.833	0.834
yeast-2 vs 8	0.738	0.729	**0.783**	0.737	0.749	0.768	0.735	0.741	0.726
yeast4	0.629	0.630	0.671	0.630	0.666	0.651	0.666	0.642	**0.681**
yeast5	0.801	0.811	0.860	0.812	**0.869**	0.838	0.838	0.844	0.867
yeast6	0.708	0.704	0.735	0.727	0.744	0.750	0.748	0.752	**0.772**
ecoli-0-1-4-7 vs 2-3-5-6	0.783	0.774	0.800	0.797	0.799	0.799	0.810	**0.824**	0.816
ecoli-0-1 vs 2-3-5	0.763	0.778	0.783	0.775	0.798	0.796	0.827	**0.836**	**0.836**
ecoli-0-2-6-7 vs 3-5	0.791	0.770	0.801	0.738	0.802	0.803	0.810	**0.827**	0.794
ecoli-0-6-7 vs 3-5	0.812	0.793	0.804	0.820	0.812	0.812	0.818	**0.830**	0.810
ecoli-0-6-7 vs 5	0.823	0.798	0.826	0.770	0.770	0.860	0.846	0.856	**0.870**
yeast-0-2-5-6 vs 3-7-8-9	0.716	0.707	0.708	0.700	0.700	0.712	0.721	0.720	**0.739**
yeast-0-3-5-9 vs 7-8	0.632	0.618	0.612	0.610	0.589	**0.639**	0.629	0.624	0.625
abalone-17 vs 7-8-9-10	0.647	0.614	0.652	0.624	0.677	0.644	0.632	0.645	**0.705**
abalone-19 vs 10-11-12-13	0.538	0.523	0.569	0.520	0.562	0.527	0.551	0.555	**0.600**
abalone-20 vs 8-9-10	0.628	0.611	**0.697**	0.635	**0.697**	0.654	0.634	0.645	0.664
flare-F	0.57	0.598	0.584	0.593	0.574	**0.616**	0.593	0.587	0.588
kr-vs-k-zero vs eight	0.976	0.914	0.972	0.905	0.938	0.954	**0.979**	0.953	**0.979**
poker-8-9 vs 5	0.546	0.544	0.578	0.537	0.566	0.544	0.56	0.531	**0.584**
poker-8-9 vs 6	0.581	0.535	0.650	0.595	0.659	0.701	**0.925**	0.921	0.898
poker-8 vs 6	0.601	0.602	0.691	0.609	0.704	0.699	**0.813**	0.812	0.812
winequality-red-4	0.545	0.533	0.589	0.548	0.542	0.574	0.547	0.560	**0.600**
winequality-white-3-9 vs 5	0.577	0.529	0.541	0.579	0.532	0.554	**0.586**	0.579	0.582
winequality-white-3 vs 7	0.603	0.553	0.552	0.613	0.539	0.567	**0.621**	0.601	0.572
ecoli1	0.816	0.822	0.832	0.837	0.842	0.829	0.837	**0.843**	**0.843**
ecoli2	0.829	0.850	0.850	0.824	0.848	**0.860**	**0.860**	0.853	0.851
ecoli3	0.755	0.694	0.754	0.763	0.776	0.757	0.796	0.808	**0.813**
glass0	0.741	0.775	0.771	0.764	0.765	0.768	0.789	**0.797**	0.771
glass1	0.730	0.722	0.719	0.733	0.685	0.729	0.731	**0.739**	0.728
haberman	0.565	0.552	0.570	0.576	0.571	**0.610**	0.600	0.582	0.586
pima	0.674	0.668	0.676	0.663	0.652	**0.684**	0.659	0.66	0.669
yeast3	0.827	0.825	0.849	0.832	0.858	0.845	0.856	0.856	**0.865**

Table 8.10: *F1 score results of compared algorithms for* CART *model.*

DATASET	Bare classifier	ROS	SMOTE	Borderline SMOTE	ADASYN	CCR	balanced	best precision	best recall
page-blocks-1-3 vs 4	0.865	**0.954**	0.944	0.790	0.943	0.879	0.875	0.897	0.880
yeast-0-5-6-7-9 vs 4	0.376	0.381	0.383	0.408	**0.420**	0.371	0.392	0.416	0.403
yeast-1-2-8-9 vs 7	0.218	0.177	0.130	0.204	0.136	**0.236**	0.195	0.191	0.210
yeast-1-4-5-8 vs 7	0.124	0.106	0.105	0.108	0.117	0.115	0.108	0.120	**0.136**
yeast-1 vs 7	0.339	0.279	0.222	0.256	0.230	0.312	0.355	**0.356**	0.321
yeast-2 vs 4	0.681	0.682	0.714	0.639	0.694	**0.733**	0.682	0.699	0.687
yeast-2 vs 8	0.486	0.480	0.474	0.477	0.412	**0.526**	0.455	0.469	0.426
yeast4	0.268	0.282	0.275	0.242	0.267	0.309	0.298	0.265	**0.319**
yeast5	0.619	0.642	0.68	0.575	**0.696**	0.634	0.634	0.633	0.661
yeast6	0.391	0.423	0.351	0.392	0.362	0.423	**0.444**	0.434	0.415
ecoli-0-1-4-7 vs 2-3-5-6	0.590	0.608	0.560	0.607	0.573	0.611	0.626	**0.631**	0.629
ecoli-0-1 vs 2-3-5	0.602	0.637	0.604	0.630	0.613	0.607	0.675	0.688	**0.695**
ecoli-0-2-6-7 vs 3-5	0.614	0.606	0.610	0.520	0.602	0.624	0.639	**0.658**	0.618
ecoli-0-6-7 vs 3-5	**0.666**	0.634	0.599	0.623	0.591	0.663	0.638	0.644	0.655
ecoli-0-6-7 vs 5	0.704	0.673	0.663	0.584	0.576	0.742	0.723	0.719	**0.747**
yeast-0-2-5-6 vs 3-7-8-9	0.476	0.483	0.436	0.434	0.413	0.465	0.473	0.471	**0.494**
yeast-0-3-5-9 vs 7-8	0.336	0.311	0.277	0.282	0.246	**0.342**	0.322	0.313	0.318
abalone-17 vs 7-8-9-10	**0.286**	0.251	0.218	0.227	0.246	0.231	0.250	0.280	0.234
abalone-19 vs 10-11-12-13	0.084	0.070	0.085	0.053	0.082	0.069	0.094	0.097	**0.098**
abalone-20 vs 8-9-10	0.245	0.240	0.239	**0.248**	0.237	0.225	0.224	0.234	0.194
flare-F	0.189	0.182	0.194	0.189	0.176	0.200	**0.218**	0.209	0.208
kr-vs-k-zero vs eight	**0.928**	0.838	0.921	0.819	0.872	0.897	0.905	0.862	0.916
poker-8-9 vs 5	0.091	0.095	0.100	0.078	0.091	0.097	**0.115**	0.069	0.111
poker-8-9 vs 6	0.180	0.085	0.290	0.182	0.289	0.435	**0.823**	0.783	0.754
poker-8 vs 6	0.248	0.254	0.418	0.254	0.423	0.425	0.638	**0.661**	0.627
winequality-red-4	0.114	0.092	0.144	0.117	0.093	0.136	0.109	0.132	**0.149**
winequality-white-3-9 vs 5	**0.177**	0.072	0.066	0.164	0.055	0.092	0.171	0.145	0.139
winequality-white-3 vs 7	0.237	0.134	0.093	**0.256**	0.076	0.125	0.252	0.209	0.155
ecoli1	0.711	0.720	0.728	0.730	0.741	0.729	0.736	**0.747**	0.736
ecoli2	0.712	0.741	0.730	0.702	0.711	**0.757**	0.740	0.743	0.731
ecoli3	0.568	0.449	0.514	0.549	0.558	0.535	0.592	**0.617**	0.578
glass0	0.652	0.698	0.689	0.680	0.682	0.685	0.709	**0.722**	0.689
glass1	0.651	0.644	0.642	0.659	0.596	0.653	0.655	**0.665**	0.654
haberman	0.361	0.336	0.379	0.385	0.381	**0.435**	0.426	0.404	0.427
pima	0.575	0.57	0.583	0.568	0.549	**0.591**	0.561	0.563	0.571
yeast3	0.694	0.691	0.699	0.681	0.706	0.721	0.714	0.719	**0.726**

Table 8.11: *Gmean results of compared algorithms for* CART *model.*

DATASET	Bare classifier	ROS	SMOTE	Borderline SMOTE	ADASYN	CCR	balanced	best precision	best recall
page-blocks-1-3 vs 4	0.924	**0.986**	**0.986**	0.895	0.979	0.971	0.958	0.970	0.962
yeast-0-5-6-7-9 vs 4	0.605	0.606	0.652	0.654	**0.689**	0.596	0.628	0.641	0.638
yeast-1-2-8-9 vs 7	0.479	0.392	0.440	0.481	0.456	0.499	0.469	0.466	**0.501**
yeast-1-4-5-8 vs 7	0.335	0.296	0.382	0.317	**0.386**	0.326	0.317	0.334	0.367
yeast-1 vs 7	0.586	0.483	0.516	0.524	0.524	0.557	**0.621**	0.617	0.58
yeast-2 vs 4	0.813	0.800	0.851	0.802	0.844	**0.852**	0.823	0.819	0.822
yeast-2 vs 8	0.691	0.679	**0.758**	0.692	0.719	0.731	0.690	0.697	0.678
yeast4	0.517	0.512	0.604	0.522	0.591	0.558	0.596	0.557	**0.620**
yeast5	0.775	0.788	0.845	0.789	**0.858**	0.822	0.822	0.828	0.856
yeast6	0.648	0.633	0.692	0.676	0.706	0.709	0.707	0.713	**0.742**
ecoli-0-1-4-7 vs 2-3-5-6	0.760	0.746	0.786	0.775	0.782	0.782	0.791	**0.809**	0.801
ecoli-0-1 vs 2-3-5	0.718	0.750	0.758	0.743	0.781	0.776	0.810	**0.822**	**0.822**
ecoli-0-2-6-7 vs 3-5	0.767	0.740	0.784	0.696	0.785	0.784	0.788	**0.812**	0.772
ecoli-0-6-7 vs 3-5	0.794	0.773	0.791	0.807	0.802	0.795	0.803	**0.817**	0.793
ecoli-0-6-7 vs 5	0.803	0.774	0.812	0.73	0.738	0.849	0.833	0.844	**0.860**
yeast-0-2-5-6 vs 3-7-8-9	0.679	0.663	0.677	0.663	0.669	0.675	0.689	0.688	**0.714**
yeast-0-3-5-9 vs 7-8	0.554	0.529	**0.559**	0.539	0.517	0.572	0.556	0.553	0.555
abalone-17 vs 7-8-9-10	0.552	0.485	0.575	0.518	0.613	0.558	0.524	0.552	**0.665**
abalone-19 vs 10-11-12-13	0.272	0.202	0.416	0.208	0.417	0.260	0.336	0.340	**0.508**
abalone-20 vs 8-9-10	0.510	0.459	**0.633**	0.519	0.630	0.563	0.521	0.541	0.583
flare-F	0.395	0.478	0.426	0.470	0.406	**0.528**	0.454	0.442	0.444
kr-vs-k-zero vs eight	0.974	0.905	0.970	0.898	0.933	0.951	**0.978**	0.905	**0.978**
poker-8-9 vs 5	0.298	0.250	**0.408**	0.259	0.384	0.229	0.336	0.247	**0.408**
poker-8-9 vs 6	0.339	0.191	0.535	0.436	0.529	0.608	**0.920**	0.913	0.887
poker-8 vs 6	0.355	0.335	0.579	0.403	0.612	0.578	**0.781**	**0.781**	0.777
winequality-red-4	0.333	0.262	0.480	0.348	0.384	0.430	0.357	0.388	**0.508**
winequality-white-3-9 vs 5	0.397	0.201	0.312	0.402	0.308	0.336	**0.423**	0.380	0.405
winequality-white-3 vs 7	0.456	0.300	0.367	**0.479**	0.235	0.321	0.463	0.429	0.378
ecoli1	0.809	0.814	0.827	0.832	0.838	0.822	0.833	0.839	**0.840**
ecoli2	0.820	0.842	0.842	0.813	0.842	**0.854**	**0.854**	0.846	0.844
ecoli3	0.726	0.646	0.733	0.741	0.759	0.734	0.779	0.793	**0.804**
glass0	0.737	0.771	0.767	0.761	0.762	0.764	0.784	**0.793**	0.767
glass1	0.724	0.717	0.715	0.730	0.679	0.726	0.728	**0.736**	0.727
haberman	0.526	0.503	0.545	0.550	0.547	**0.593**	0.586	0.568	0.583
pima	0.666	0.662	0.672	0.659	0.644	**0.678**	0.653	0.656	0.662
yeast3	0.816	0.813	0.843	0.823	0.852	0.837	0.850	0.850	**0.860**

C Additional results for analysis of the fitness calculation protocols

Table 8.12: *Average precision scores for* CART *classifier*

DATASET	hold-out			train			cv		
	precision	recall	balanced	precision	recall	balanced	precision	recall	balanced
adult	0.611	0.600	0.609	0.611	0.591	0.602	**0.618**	0.606	0.618
bank_additional	0.516	0.508	0.517	0.520	0.498	0.519	0.526	0.517	**0.527**
page-blocks0	0.815	0.795	0.794	0.825	0.794	0.807	**0.845**	0.794	0.808
glass1	0.602	0.618	0.617	0.621	0.621	0.621	**0.678**	0.533	0.643
glass0	0.659	0.668	0.672	0.683	0.683	0.683	0.717	0.587	**0.744**
ecoli-0-6-7_vs_5	0.807	0.807	0.807	0.804	0.804	0.804	**0.870**	0.757	0.796
ecoli-0-6-7_vs_3-5	**0.815**	0.815	0.815	0.687	0.687	0.687	0.784	0.753	0.751
ecoli-0-2-6-7_vs_3-5	0.724	0.724	0.724	0.671	0.671	0.671	**0.811**	0.709	0.760
ecoli-0-1_vs_2-3-5	0.689	0.689	0.689	0.691	0.691	0.691	**0.720**	0.543	0.621
haberman	0.348	0.370	0.351	0.359	0.357	0.358	0.362	0.371	**0.392**
ecoli1	0.743	0.748	0.748	0.725	0.720	0.722	**0.908**	0.691	0.787
ecoli2	0.773	0.773	0.773	0.729	0.729	0.729	**0.782**	0.684	0.770
ecoli3	0.563	0.546	0.563	0.523	0.523	0.523	**0.727**	0.481	0.613
ecoli-0-1-4-7_vs_2-3-5-6	0.535	0.535	0.535	0.602	0.613	0.613	**0.729**	0.606	0.704
yeast-1_vs_7	0.307	0.255	0.234	0.307	0.307	0.307	**0.453**	0.304	0.287
page-blocks-1-3_vs_4	0.799	0.799	0.799	0.909	0.909	0.909	0.836	0.891	**0.930**
yeast-2_vs_8	0.685	0.685	0.685	0.472	0.486	0.486	**0.831**	0.556	0.600
yeast-0-3-5-9_vs_7-8	0.349	0.333	0.345	0.266	0.281	0.282	**0.532**	0.307	0.398
yeast-2_vs_4	0.713	0.722	0.723	0.708	0.708	0.708	**0.846**	0.635	0.694
yeast-0-5-6-7-9_vs_4	0.474	0.455	0.474	0.363	0.363	0.356	**0.521**	0.358	0.485
yeast-1-4-5-8_vs_7	**0.124**	0.117	0.108	0.099	0.114	0.114	0.099	0.074	0.082
pima	0.576	0.551	0.555	0.575	0.571	0.580	**0.617**	0.562	0.591
winequality-white-3_vs_7	**0.553**	0.553	0.553	0.259	0.259	0.259	0.342	0.319	0.320
yeast-1-2-8-9_vs_7	0.225	0.226	0.226	0.235	0.231	0.230	**0.414**	0.168	0.194
yeast-0-2-5-6_vs_3-7-8-9	0.524	0.466	0.477	0.516	0.428	0.458	**0.679**	0.469	0.569
flare-F	0.213	0.204	0.204	0.212	0.186	0.211	0.274	**0.303**	0.266
kr-vs-k-zero_vs_eight	0.715	0.715	0.715	**0.898**	0.898	0.898	0.873	0.749	0.832
poker-8_vs_6	0.714	0.714	0.714	0.838	0.744	0.744	**1.000**	0.872	0.872
winequality-white-3-9_vs_5	0.093	0.100	0.094	**0.243**	0.243	0.243	0.159	0.146	0.166
yeast3	0.734	0.730	0.735	0.673	0.650	0.646	**0.800**	0.695	0.734
poker-8-9_vs_6	0.657	0.668	0.657	0.668	0.668	0.668	**0.879**	0.630	0.630
yeast6	0.352	0.326	0.352	0.392	0.379	0.379	0.397	0.365	**0.432**
yeast5	0.633	0.633	0.633	0.620	0.620	0.620	0.608	0.625	**0.674**
yeast4	0.303	0.279	0.279	0.261	0.260	0.268	**0.447**	0.262	0.306
winequality-red-4	0.139	0.088	0.088	0.112	0.092	0.116	**0.180**	0.120	0.118
abalone-19_vs_10-11-12-13	0.045	0.058	0.058	0.112	0.101	**0.122**	0.086	0.053	0.060
abalone-20_vs_8-9-10	0.171	0.159	0.162	**0.263**	0.234	0.242	0.244	0.234	0.226
poker-8-9_vs_5	0.080	0.080	0.080	0.135	0.151	0.120	**0.247**	0.057	0.076
abalone-17_vs_7-8-9-10	0.285	0.220	0.255	**0.291**	0.264	0.268	0.272	0.216	0.224

Table 8.13: *Average recall scores for* CART *classifier*

DATASET	hold-out			train			cv		
	precision	recall	balanced	precision	recall	balanced	precision	recall	balanced
adult	0.621	0.635	0.624	0.602	0.639	0.620	0.626	**0.643**	0.627
bank_additional	0.538	**0.556**	0.540	0.511	0.551	0.529	0.541	0.552	0.541
page-blocks0	0.781	0.801	0.799	0.759	0.814	0.804	0.771	**0.841**	0.818
glass1	0.603	0.618	0.613	0.658	0.658	0.658	0.508	**0.776**	0.634
glass0	0.674	0.680	0.671	0.734	0.734	0.734	0.660	**0.857**	0.694
ecoli-0-6-7_vs_5	0.660	0.660	0.660	0.690	0.690	0.690	0.620	**0.760**	0.760
ecoli-0-6-7_vs_3-5	0.591	0.591	0.591	0.655	0.655	0.655	0.582	**0.727**	0.718
ecoli-0-2-6-7_vs_3-5	0.545	0.545	0.545	0.673	0.673	0.673	0.618	0.682	**0.691**
ecoli-0-1_vs_2-3-5	0.542	0.542	0.542	0.583	0.583	0.583	0.450	**0.650**	0.567
haberman	0.318	0.370	0.323	0.370	0.375	0.378	0.276	**0.635**	0.443
ecoli1	0.770	0.787	0.787	0.731	0.746	0.744	0.608	**0.881**	0.754
ecoli2	0.692	0.692	0.692	0.719	0.719	0.719	0.654	**0.804**	0.742
ecoli3	0.548	0.554	0.548	0.521	0.521	0.521	0.408	**0.738**	0.585
ecoli-0-1-4-7_vs_2-3-5-6	0.587	0.587	0.587	0.628	0.634	0.634	0.476	0.657	**0.698**
yeast-1_vs_7	0.267	0.313	0.273	**0.420**	0.420	0.420	0.247	0.420	0.420
page-blocks-1-3_vs_4	0.871	0.871	0.871	0.936	0.936	0.936	0.829	**0.993**	0.986
yeast-2_vs_8	**0.590**	0.590	0.590	0.500	0.530	0.530	0.550	0.560	0.560
yeast-0-3-5-9_vs_7-8	0.272	0.320	0.300	0.292	0.316	0.312	0.244	**0.416**	0.400
yeast-2_vs_4	0.560	0.634	0.626	0.679	0.679	0.679	0.564	**0.820**	0.718
yeast-0-5-6-7-9_vs_4	0.399	0.399	0.399	0.389	0.424	0.404	0.366	**0.550**	0.479
yeast-1-4-5-8_vs_7	0.100	0.127	0.107	0.107	0.120	0.120	0.107	**0.160**	0.120
pima	0.505	0.565	0.531	0.526	0.621	0.569	0.565	**0.707**	0.614
winequality-white-3_vs_7	0.220	0.220	0.220	0.200	0.200	0.200	0.250	0.290	**0.290**
yeast-1-2-8-9_vs_7	0.227	0.227	0.227	0.287	**0.300**	0.293	0.153	0.233	0.220
yeast-0-2-5-6_vs_3-7-8-9	0.479	0.469	0.469	0.465	0.513	0.477	0.506	**0.616**	0.554
flare-F	0.236	0.250	0.250	0.164	0.267	0.234	0.183	**0.506**	0.332
kr-vs-k-zero_vs_eight	0.792	0.792	0.792	0.917	0.917	0.917	0.605	**0.931**	0.858
poker-8_vs_6	0.281	0.281	0.281	**0.642**	0.604	0.604	0.342	0.364	0.364
winequality-white-3-9_vs_5	0.120	0.137	0.128	**0.216**	0.216	0.216	0.145	0.215	0.190
yeast3	0.699	0.719	0.720	0.658	0.692	0.651	0.669	**0.875**	0.755
poker-8-9_vs_6	0.326	0.342	0.326	**0.629**	0.629	0.629	0.446	0.361	0.361
yeast6	0.358	0.369	0.358	0.463	0.457	0.457	0.413	**0.557**	0.533
yeast5	0.605	0.605	0.605	0.564	0.564	0.564	0.555	**0.768**	0.691
yeast4	0.247	0.247	0.247	0.310	0.310	0.314	0.271	**0.456**	0.302
winequality-red-4	0.114	0.098	0.098	0.105	0.116	0.131	0.094	**0.185**	0.143
abalone-19_vs_10-11-12-13	0.038	0.050	0.050	0.119	**0.150**	0.131	0.081	0.062	0.069
abalone-20_vs_8-9-10	0.192	0.215	0.215	0.277	0.285	0.300	0.208	**0.408**	0.292
poker-8-9_vs_5	0.072	0.072	0.072	0.160	**0.193**	0.160	0.096	0.088	0.113
abalone-17_vs_7-8-9-10	0.228	0.224	0.214	0.248	0.290	0.255	0.169	**0.300**	0.245

Table 8.14: *Average* BAC *scores for* CART *classifier*

DATASET	hold-out			train			cv		
	precision	recall	balanced	precision	recall	balanced	precision	recall	balanced
adult	0.747	0.750	0.748	0.740	0.749	0.745	0.752	**0.755**	0.752
bank_additional	0.737	**0.744**	0.738	0.726	0.740	0.733	0.739	0.743	0.740
page-blocks0	0.880	0.889	0.887	0.870	0.895	0.891	0.877	**0.908**	0.898
glass1	0.686	0.699	0.697	**0.718**	0.718	0.718	0.684	0.700	0.711
glass0	0.751	0.755	0.754	0.780	0.780	0.780	0.764	0.770	**0.784**
ecoli-0-6-7_vs_5	0.816	0.816	0.816	0.833	0.833	0.833	0.804	0.866	**0.868**
ecoli-0-6-7_vs_3-5	0.787	0.787	0.787	0.807	0.807	0.807	0.781	**0.849**	0.844
ecoli-0-2-6-7_vs_3-5	0.752	0.752	0.752	0.815	0.815	0.815	0.800	0.824	**0.832**
ecoli-0-1_vs_2-3-5	0.756	0.756	0.756	0.777	0.777	0.777	0.715	**0.790**	0.765
haberman	0.555	0.575	0.555	0.566	0.566	0.567	0.556	**0.620**	0.597
ecoli1	0.843	0.852	0.852	0.823	0.828	0.827	0.794	**0.880**	0.845
ecoli2	0.827	0.827	0.827	0.834	0.834	0.834	0.809	**0.865**	0.849
ecoli3	0.749	0.749	0.749	0.732	0.732	0.732	0.692	**0.821**	0.771
ecoli-0-1-4-7_vs_2-3-5-6	0.767	0.767	0.767	0.794	0.798	0.798	0.730	0.806	**0.834**
yeast-1_vs_7	0.611	0.625	0.607	**0.676**	0.676	0.676	0.607	0.674	0.671
page-blocks-1-3_vs_4	0.928	0.928	0.928	0.965	0.965	0.965	0.909	**0.991**	0.990
yeast-2_vs_8	**0.786**	0.786	0.786	0.736	0.751	0.751	0.772	0.769	0.770
yeast-0-3-5-9_vs_7-8	0.606	0.623	0.617	0.601	0.613	0.612	0.609	0.655	**0.661**
yeast-2_vs_4	0.767	0.803	0.799	0.824	0.824	0.824	0.776	**0.883**	0.840
yeast-0-5-6-7-9_vs_4	0.675	0.672	0.675	0.658	0.673	0.663	0.664	**0.720**	0.711
yeast-1-4-5-8_vs_7	0.526	**0.538**	0.530	0.530	0.537	0.537	0.533	0.535	0.534
pima	0.653	0.659	0.651	0.658	0.684	0.673	0.686	**0.705**	0.693
winequality-white-3_vs_7	0.606	0.606	0.606	0.593	0.593	0.593	0.620	0.635	**0.636**
yeast-1-2-8-9_vs_7	0.600	0.600	0.600	0.627	**0.632**	0.629	0.571	0.595	0.594
yeast-0-2-5-6_vs_3-7-8-9	0.715	0.704	0.704	0.708	0.718	0.706	0.738	**0.769**	0.753
flare-F	0.601	0.605	0.605	0.569	0.607	0.598	0.582	**0.728**	0.648
kr-vs-k-zero_vs_eight	0.893	0.893	0.893	0.958	0.958	0.958	0.801	**0.962**	0.926
poker-8_vs_6	0.638	0.638	0.638	**0.819**	0.799	0.799	0.671	0.681	0.681
winequality-white-3-9_vs_5	0.551	0.559	0.555	**0.601**	0.601	0.601	0.567	0.597	0.587
yeast3	0.834	0.843	0.844	0.809	0.823	0.803	0.824	**0.914**	0.860
poker-8-9_vs_6	0.659	0.668	0.659	**0.812**	0.812	0.812	0.722	0.677	0.677
yeast6	0.670	0.674	0.670	0.722	0.719	0.719	0.699	**0.766**	0.757
yeast5	0.797	0.797	0.797	0.777	0.777	0.777	0.772	**0.877**	0.840
yeast4	0.613	0.612	0.612	0.639	0.639	0.642	0.628	**0.705**	0.638
winequality-red-4	0.543	0.531	0.531	0.538	0.538	0.548	0.538	**0.568**	0.553
abalone-19_vs_10-11-12-13	0.513	0.518	0.518	0.550	**0.561**	0.556	0.533	0.517	0.522
abalone-20_vs_8-9-10	0.589	0.600	0.600	0.633	0.635	0.643	0.599	**0.695**	0.639
poker-8-9_vs_5	0.530	0.530	0.530	0.573	**0.589**	0.572	0.544	0.535	0.548
abalone-17_vs_7-8-9-10	0.607	0.602	0.599	0.616	0.634	0.618	0.579	**0.635**	0.611

Table 8.15: *Average F1 score scores for* CART *classifier*

DATASET	hold-out			train			cv		
	precision	recall	balanced	precision	recall	balanced	precision	recall	balanced
adult	0.616	0.617	0.616	0.607	0.614	0.611	0.622	**0.623**	0.623
bank_additional	0.526	0.531	0.528	0.516	0.523	0.524	0.533	**0.534**	0.533
page-blocks0	0.796	0.796	0.795	0.789	0.803	0.804	0.805	**0.816**	0.812
glass1	0.594	0.608	0.606	**0.636**	0.636	0.636	0.577	0.631	0.630
glass0	0.663	0.669	0.668	0.701	0.701	0.701	0.682	0.689	**0.710**
ecoli-0-6-7_vs_5	0.688	0.688	0.688	0.722	0.722	0.722	0.701	0.741	**0.757**
ecoli-0-6-7_vs_3-5	0.668	0.668	0.668	0.656	0.656	0.656	0.656	**0.734**	0.727
ecoli-0-2-6-7_vs_3-5	0.591	0.591	0.591	0.659	0.659	0.659	0.688	0.684	**0.711**
ecoli-0-1_vs_2-3-5	0.589	0.589	0.589	**0.623**	0.623	0.623	0.541	0.559	0.575
haberman	0.326	0.364	0.331	0.362	0.364	0.365	0.308	**0.464**	0.412
ecoli1	0.749	0.759	0.759	0.722	0.728	0.727	0.724	**0.771**	0.763
ecoli2	0.726	0.726	0.726	0.717	0.717	0.717	0.707	0.729	**0.750**
ecoli3	0.547	0.541	0.547	0.520	0.520	0.520	0.494	0.575	**0.594**
ecoli-0-1-4-7_vs_2-3-5-6	0.548	0.548	0.548	0.613	0.622	0.622	0.557	0.615	**0.689**
yeast-1_vs_7	0.267	0.274	0.246	**0.351**	0.351	0.351	0.291	0.346	0.333
page-blocks-1-3_vs_4	0.821	0.821	0.821	0.919	0.919	0.919	0.815	0.934	**0.956**
yeast-2_vs_8	0.602	0.602	0.602	0.465	0.484	0.484	**0.648**	0.546	0.563
yeast-0-3-5-9_vs_7-8	0.301	0.321	0.316	0.274	0.293	0.292	0.327	0.351	**0.385**
yeast-2_vs_4	0.597	0.667	0.660	0.688	0.688	0.688	0.670	**0.706**	0.697
yeast-0-5-6-7-9_vs_4	0.428	0.416	0.428	0.374	0.386	0.375	0.420	0.426	**0.476**
yeast-1-4-5-8_vs_7	0.091	0.113	0.100	0.102	**0.116**	0.116	0.101	0.097	0.097
pima	0.536	0.557	0.542	0.547	0.594	0.573	0.585	**0.624**	0.601
winequality-white-3_vs_7	**0.279**	0.279	0.279	0.220	0.220	0.220	0.261	0.271	0.273
yeast-1-2-8-9_vs_7	0.215	0.216	0.216	0.250	**0.251**	0.248	0.204	0.186	0.198
yeast-0-2-5-6_vs_3-7-8-9	0.491	0.453	0.457	0.485	0.459	0.462	**0.567**	0.530	0.557
flare-F	0.214	0.218	0.218	0.180	0.215	0.219	0.211	**0.373**	0.284
kr-vs-k-zero_vs_eight	0.729	0.729	0.729	**0.905**	0.905	0.905	0.681	0.819	0.824
poker-8_vs_6	0.396	0.396	0.396	**0.716**	0.658	0.658	0.484	0.460	0.460
winequality-white-3-9_vs_5	0.104	0.114	0.107	**0.217**	0.217	0.217	0.141	0.167	0.166
yeast3	0.713	0.722	0.726	0.663	0.669	0.646	0.726	**0.774**	0.743
poker-8-9_vs_6	0.424	0.437	0.424	**0.640**	0.640	0.640	0.571	0.443	0.443
yeast6	0.339	0.327	0.339	0.418	0.409	0.409	0.395	0.435	**0.465**
yeast5	0.613	0.613	0.613	0.581	0.581	0.581	0.573	**0.679**	0.676
yeast4	0.263	0.257	0.257	0.278	0.279	0.286	0.319	**0.329**	0.295
winequality-red-4	0.116	0.089	0.089	0.106	0.101	0.120	0.110	**0.143**	0.126
abalone-19_vs_10-11-12-13	0.039	0.051	0.051	0.113	0.119	**0.125**	0.081	0.056	0.062
abalone-20_vs_8-9-10	0.174	0.176	0.178	0.265	0.252	0.263	0.219	**0.293**	0.249
poker-8-9_vs_5	0.069	0.069	0.069	0.141	**0.163**	0.133	0.118	0.068	0.089
abalone-17_vs_7-8-9-10	0.249	0.216	0.228	0.261	**0.268**	0.254	0.201	0.243	0.231

Table 8.16: *Average Gmean scores for* CART *classifier*

DATASET	hold-out			train			cv		
	precision	recall	balanced	precision	recall	balanced	precision	recall	balanced
adult	0.737	0.741	0.738	0.727	0.741	0.734	0.741	**0.747**	0.742
bank_additional	0.709	**0.719**	0.711	0.693	0.716	0.704	0.712	0.718	0.712
page-blocks0	0.875	0.884	0.883	0.863	0.891	0.886	0.871	**0.905**	0.894
glass1	0.676	0.688	0.686	**0.713**	0.713	0.713	0.659	0.694	0.703
glass0	0.745	0.749	0.747	0.775	0.775	0.775	0.755	0.758	**0.776**
ecoli-0-6-7_vs_5	0.790	0.790	0.790	0.816	0.816	0.816	0.773	0.854	**0.855**
ecoli-0-6-7_vs_3-5	0.756	0.756	0.756	0.789	0.789	0.789	0.750	**0.838**	0.833
ecoli-0-2-6-7_vs_3-5	0.716	0.716	0.716	0.798	0.798	0.798	0.774	0.807	**0.815**
ecoli-0-1_vs_2-3-5	0.717	0.717	0.717	0.747	0.747	0.747	0.657	**0.761**	0.721
haberman	0.483	0.525	0.495	0.527	0.529	0.531	0.467	**0.614**	0.572
ecoli1	0.837	0.846	0.846	0.815	0.822	0.821	0.771	**0.879**	0.837
ecoli2	0.814	0.814	0.814	0.823	0.823	0.823	0.792	**0.860**	0.841
ecoli3	0.712	0.715	0.712	0.696	0.696	0.696	0.618	**0.814**	0.744
ecoli-0-1-4-7_vs_2-3-5-6	0.740	0.740	0.740	0.774	0.779	0.779	0.670	0.788	**0.818**
yeast-1_vs_7	0.480	0.527	0.484	**0.619**	0.619	0.619	0.476	0.614	0.616
page-blocks-1-3_vs_4	0.921	0.921	0.921	0.963	0.963	0.963	0.890	**0.991**	0.990
yeast-2_vs_8	**0.756**	0.756	0.756	0.688	0.710	0.710	0.736	0.737	0.738
yeast-0-3-5-9_vs_7-8	0.501	0.539	0.526	0.510	0.532	0.529	0.484	**0.608**	0.603
yeast-2_vs_4	0.720	0.780	0.773	0.809	0.809	0.809	0.742	**0.877**	0.828
yeast-0-5-6-7-9_vs_4	0.612	0.610	0.612	0.597	0.618	0.607	0.585	**0.694**	0.667
yeast-1-4-5-8_vs_7	0.266	0.319	0.277	0.275	0.293	0.293	0.292	**0.365**	0.293
pima	0.634	0.651	0.639	0.643	0.681	0.664	0.672	**0.703**	0.687
winequality-white-3_vs_7	0.442	0.442	0.442	0.438	0.438	0.438	0.432	0.519	**0.520**
yeast-1-2-8-9_vs_7	0.453	0.453	0.453	0.520	**0.531**	0.525	0.381	0.457	0.448
yeast-0-2-5-6_vs_3-7-8-9	0.669	0.655	0.656	0.662	0.684	0.666	0.697	**0.753**	0.725
flare-F	0.453	0.474	0.474	0.392	0.494	0.469	0.390	**0.688**	0.541
kr-vs-k-zero_vs_eight	0.879	0.879	0.879	0.954	0.954	0.954	0.766	**0.958**	0.920
poker-8_vs_6	0.463	0.463	0.463	**0.750**	0.714	0.714	0.563	0.579	0.579
winequality-white-3-9_vs_5	0.296	0.317	0.305	**0.450**	0.450	0.450	0.299	0.442	0.412
yeast3	0.822	0.833	0.834	0.793	0.812	0.788	0.808	**0.913**	0.853
poker-8-9_vs_6	0.502	0.519	0.502	**0.744**	0.744	0.744	0.663	0.590	0.590
yeast6	0.565	0.578	0.565	0.673	0.668	0.668	0.625	**0.733**	0.720
yeast5	0.769	0.769	0.769	0.739	0.739	0.739	0.721	**0.866**	0.824
yeast4	0.482	0.485	0.485	0.535	0.541	0.544	0.509	**0.655**	0.537
winequality-red-4	0.322	0.291	0.291	0.292	0.298	0.342	0.267	**0.416**	0.367
abalone-19_vs_10-11-12-13	0.134	0.169	0.169	0.335	**0.357**	0.331	0.245	0.218	0.214
abalone-20_vs_8-9-10	0.420	0.443	0.444	0.516	0.518	0.537	0.426	**0.624**	0.528
poker-8-9_vs_5	0.221	0.221	0.221	0.361	**0.418**	0.381	0.272	0.259	0.311
abalone-17_vs_7-8-9-10	0.460	0.455	0.447	0.484	0.524	0.487	0.394	**0.535**	0.486

Table 8.17: *Average precision scores for* KNN *classifier*

DATASET	hold-out			train			cv		
	precision	recall	balanced	precision	recall	balanced	precision	recall	balanced
adult	0.649	0.645	0.645	0.657	0.631	0.632	**0.663**	0.640	0.640
bank_additional	0.586	0.581	0.586	0.586	0.579	0.581	**0.588**	0.580	0.588
page-blocks0	0.852	0.849	0.849	0.864	0.815	0.815	**0.892**	0.798	0.798
glass1	0.662	0.646	0.671	0.684	0.610	0.640	**0.732**	0.488	0.611
glass0	0.615	0.575	0.585	0.636	0.602	0.665	**0.681**	0.520	0.617
ecoli-0-6-7_vs_5	0.851	0.851	0.851	**0.854**	0.836	0.836	0.796	0.781	0.739
ecoli-0-6-7_vs_3-5	0.812	0.812	0.812	**0.862**	0.850	0.850	0.794	0.722	0.755
ecoli-0-2-6-7_vs_3-5	0.824	0.824	0.824	0.885	**0.887**	0.887	0.872	0.758	0.770
ecoli-0-1_vs_2-3-5	0.945	0.945	0.945	**0.949**	0.949	0.949	0.862	0.848	0.848
haberman	0.443	0.437	0.437	**0.452**	0.383	0.400	0.434	0.307	0.386
ecoli1	0.714	0.708	0.708	0.826	0.719	0.768	**0.905**	0.725	0.812
ecoli2	0.808	0.808	0.808	0.848	0.849	0.848	**0.896**	0.848	0.859
ecoli3	0.538	0.538	0.538	**0.631**	0.537	0.619	0.621	0.556	0.621
ecoli-0-1-4-7_vs_2-3-5-6	0.877	0.877	0.877	**0.907**	0.858	0.869	0.828	0.792	0.797
yeast-1_vs_7	0.434	0.434	0.434	**0.781**	0.734	0.734	0.642	0.445	0.445
page-blocks-1-3_vs_4	0.703	0.703	0.703	0.835	**0.838**	0.838	0.788	0.695	0.741
yeast-2_vs_8	0.649	0.649	0.649	0.822	**0.922**	0.922	0.912	0.912	0.912
yeast-0-3-5-9_vs_7-8	**0.637**	0.637	0.637	0.540	0.516	0.516	0.559	0.400	0.394
yeast-2_vs_4	**0.890**	0.890	0.890	0.882	0.882	0.882	0.856	0.809	0.809
yeast-0-5-6-7-9_vs_4	**0.541**	0.541	0.541	0.537	0.532	0.532	0.535	0.430	0.458
yeast-1-4-5-8_vs_7	0.153	0.153	0.153	0.142	0.142	0.142	0.144	**0.156**	0.156
pima	0.633	0.630	0.630	0.633	0.567	0.594	**0.706**	0.548	0.610
winequality-white-3_vs_7	**0.000**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
yeast-1-2-8-9_vs_7	0.233	0.233	0.233	0.578	**0.603**	0.603	0.335	0.340	0.340
yeast-0-2-5-6_vs_3-7-8-9	0.672	0.679	0.679	**0.753**	0.697	0.697	0.722	0.643	0.640
flare-F	0.309	0.356	0.356	0.141	0.301	0.311	**0.420**	0.280	0.284
kr-vs-k-zero_vs_eight	0.730	0.730	0.730	0.744	0.742	0.742	**0.759**	0.687	0.687
poker-8_vs_6	0.400	0.400	0.400	**0.900**	0.900	0.900	0.567	0.567	0.567
winequality-white-3-9_vs_5	0.000	0.000	0.000	**0.133**	0.133	0.133	0.075	0.075	0.075
yeast3	0.822	0.821	0.821	0.786	0.770	0.769	**0.834**	0.748	0.753
poker-8-9_vs_6	0.340	0.340	0.340	**0.950**	0.950	0.950	0.890	0.890	0.890
yeast6	0.568	0.568	0.568	**0.606**	0.537	0.551	0.583	0.509	0.529
yeast5	0.682	0.671	0.671	0.690	0.602	0.633	**0.778**	0.550	0.638
yeast4	0.440	0.440	0.440	0.504	0.491	0.491	**0.526**	0.344	0.353
winequality-red-4	0.201	0.201	0.201	0.189	0.149	0.149	**0.225**	0.197	0.197
abalone-19_vs_10-11-12-13	**0.000**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
abalone-20_vs_8-9-10	0.000	0.000	0.000	**0.300**	0.250	0.250	0.133	0.133	0.133
poker-8-9_vs_5	**0.000**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
abalone-17_vs_7-8-9-10	0.150	0.150	0.150	**0.388**	0.367	0.367	0.350	0.355	0.355

Table 8.18: *Average recall scores for* KNN *classifier*

DATASET	hold-out			train			cv		
	precision	recall	balanced	precision	recall	balanced	precision	recall	balanced
adult	0.567	0.577	0.577	0.556	0.608	0.608	0.574	**0.621**	0.621
bank_additional	0.372	0.381	0.372	0.374	0.399	0.385	0.392	**0.403**	0.392
page-blocks0	0.689	0.691	0.691	0.695	0.763	0.763	0.662	**0.776**	0.776
glass1	0.521	0.589	0.558	0.542	0.721	0.671	0.392	**0.884**	0.692
glass0	0.554	0.606	0.591	0.523	0.766	0.671	0.514	**0.926**	0.683
ecoli-0-6-7_vs_5	0.570	0.570	0.570	0.720	0.740	0.740	0.740	**0.780**	0.780
ecoli-0-6-7_vs_3-5	0.564	0.564	0.564	0.655	0.655	0.655	0.700	**0.709**	0.700
ecoli-0-2-6-7_vs_3-5	0.545	0.545	0.545	0.618	0.618	0.618	0.700	**0.718**	0.709
ecoli-0-1_vs_2-3-5	0.675	0.675	0.675	0.675	0.675	0.675	**0.725**	0.725	0.725
haberman	0.282	0.294	0.294	0.215	0.383	0.368	0.171	**0.726**	0.413
ecoli1	0.746	0.748	0.748	0.671	0.789	0.704	0.567	**0.816**	0.725
ecoli2	0.858	0.858	0.858	0.835	0.877	0.865	0.765	**0.892**	0.869
ecoli3	0.484	0.484	0.484	0.433	0.715	0.674	0.492	**0.841**	0.713
ecoli-0-1-4-7_vs_2-3-5-6	0.673	0.673	0.673	0.627	0.723	0.723	0.737	**0.758**	0.758
yeast-1_vs_7	0.113	0.113	0.113	0.227	0.220	0.220	0.273	**0.307**	0.307
page-blocks-1-3_vs_4	0.457	0.464	0.464	0.771	0.757	0.757	0.586	**0.814**	0.721
yeast-2_vs_8	0.340	0.340	0.340	0.440	**0.550**	0.550	0.550	0.550	0.550
yeast-0-3-5-9_vs_7-8	0.240	0.240	0.240	0.184	0.292	0.292	0.304	**0.480**	0.464
yeast-2_vs_4	0.635	0.635	0.635	0.678	0.678	0.678	0.695	**0.738**	0.738
yeast-0-5-6-7-9_vs_4	0.333	0.333	0.333	0.239	0.310	0.310	0.254	**0.502**	0.455
yeast-1-4-5-8_vs_7	0.033	0.033	0.033	0.033	0.033	0.033	0.053	**0.060**	0.060
pima	0.530	0.546	0.546	0.474	0.656	0.592	0.405	**0.790**	0.642
winequality-white-3_vs_7	**0.000**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
yeast-1-2-8-9_vs_7	0.027	0.027	0.027	0.087	**0.093**	0.093	0.087	0.093	0.093
yeast-0-2-5-6_vs_3-7-8-9	0.358	0.404	0.404	0.354	0.509	0.509	0.364	0.586	**0.588**
flare-F	0.103	0.130	0.130	0.061	0.294	0.276	0.237	**0.465**	0.455
kr-vs-k-zero_vs_eight	0.546	0.546	0.546	0.607	0.584	0.584	0.634	**0.648**	0.648
poker-8_vs_6	0.057	0.057	0.057	**0.175**	0.175	0.175	0.126	0.126	0.126
winequality-white-3-9_vs_5	0.000	0.000	0.000	0.016	0.016	0.016	**0.017**	0.017	0.017
yeast3	0.659	0.665	0.665	0.631	0.682	0.682	0.630	**0.753**	0.751
poker-8-9_vs_6	0.047	0.047	0.047	**0.247**	0.247	0.247	0.217	0.217	0.217
yeast6	0.470	0.470	0.470	0.425	0.441	0.441	0.532	**0.658**	0.618
yeast5	0.473	0.486	0.486	0.414	0.641	0.618	0.405	**0.818**	0.741
yeast4	0.118	0.118	0.118	0.176	0.246	0.246	0.207	**0.395**	0.372
winequality-red-4	0.023	0.023	0.023	0.030	0.030	0.030	0.064	**0.068**	0.068
abalone-19_vs_10-11-12-13	**0.000**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
abalone-20_vs_8-9-10	0.000	0.000	0.000	0.038	**0.054**	0.054	0.015	0.015	0.015
poker-8-9_vs_5	**0.000**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
abalone-17_vs_7-8-9-10	0.007	0.007	0.007	**0.055**	0.055	0.055	0.048	0.055	0.055

Table 8.19: *Average* BAC *scores for* KNN *classifier*

DATASET	hold-out			train			cv		
	precision	recall	balanced	precision	recall	balanced	precision	recall	balanced
adult	0.735	0.738	0.738	0.732	0.748	0.748	0.741	0.755	**0.755**
bank_additional	0.669	0.673	0.669	0.670	0.681	0.675	0.678	**0.683**	0.679
page-blocks0	0.837	0.839	0.839	0.841	0.871	0.871	0.827	**0.877**	0.877
glass1	0.682	0.701	0.695	0.700	0.729	**0.730**	0.654	0.678	0.723
glass0	0.691	0.683	0.684	0.688	**0.756**	0.751	0.697	0.754	0.738
ecoli-0-6-7_vs_5	0.779	0.779	0.779	0.852	0.861	0.861	0.859	**0.878**	0.876
ecoli-0-6-7_vs_3-5	0.774	0.774	0.774	0.820	0.820	0.820	**0.838**	0.834	0.836
ecoli-0-2-6-7_vs_3-5	0.763	0.763	0.763	0.803	0.803	0.803	0.843	**0.844**	0.840
ecoli-0-1_vs_2-3-5	0.835	0.835	0.835	0.835	0.835	0.835	**0.855**	0.854	0.854
haberman	0.576	0.577	0.577	0.555	0.577	0.583	0.545	0.567	**0.588**
ecoli1	0.825	0.825	0.825	0.813	0.848	0.820	0.773	**0.861**	0.836
ecoli2	0.909	0.909	0.909	0.903	0.923	0.918	0.874	**0.931**	0.921
ecoli3	0.718	0.718	0.718	0.700	0.820	0.812	0.728	**0.881**	0.831
ecoli-0-1-4-7_vs_2-3-5-6	0.831	0.831	0.831	0.810	0.856	0.856	0.861	0.869	**0.869**
yeast-1_vs_7	0.552	0.552	0.552	0.610	0.606	0.606	0.631	**0.637**	0.637
page-blocks-1-3_vs_4	0.722	0.725	0.725	0.880	0.873	0.873	0.787	**0.896**	0.852
yeast-2_vs_8	0.669	0.669	0.669	0.719	**0.774**	0.774	0.774	0.774	0.774
yeast-0-3-5-9_vs_7-8	0.610	0.610	0.610	0.584	0.631	0.631	0.638	**0.699**	0.691
yeast-2_vs_4	0.813	0.813	0.813	0.834	0.834	0.834	0.841	**0.859**	0.859
yeast-0-5-6-7-9_vs_4	0.650	0.650	0.650	0.607	0.639	0.639	0.614	**0.715**	0.698
yeast-1-4-5-8_vs_7	0.514	0.514	0.514	0.513	0.512	0.512	0.519	**0.520**	0.520
pima	0.682	0.686	0.686	0.662	0.693	0.686	0.656	**0.720**	0.709
winequality-white-3_vs_7	**0.500**	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
yeast-1-2-8-9_vs_7	0.513	0.513	0.513	0.542	0.545	0.545	0.542	**0.545**	0.545
yeast-0-2-5-6_vs_3-7-8-9	0.669	0.691	0.691	0.670	0.742	0.742	0.674	0.774	**0.775**
flare-F	0.546	0.559	0.559	0.525	0.632	0.624	0.611	**0.707**	0.703
kr-vs-k-zero_vs_eight	0.770	0.770	0.770	0.801	0.790	0.790	0.815	**0.821**	0.821
poker-8_vs_6	0.528	0.528	0.528	**0.588**	0.588	0.588	0.563	0.563	0.563
winequality-white-3-9_vs_5	0.500	0.500	0.500	0.508	0.508	0.508	**0.508**	0.508	0.508
yeast3	0.821	0.824	0.824	0.805	0.828	0.828	0.807	**0.861**	0.860
poker-8-9_vs_6	0.523	0.523	0.523	**0.623**	0.623	0.623	0.608	0.608	0.608
yeast6	0.730	0.730	0.730	0.709	0.716	0.716	0.761	**0.821**	0.802
yeast5	0.732	0.739	0.739	0.703	0.814	0.803	0.700	**0.898**	0.863
yeast4	0.556	0.556	0.556	0.585	0.618	0.618	0.600	**0.684**	0.673
winequality-red-4	0.510	0.510	0.510	0.513	0.512	0.512	0.528	**0.529**	0.529
abalone-19_vs_10-11-12-13	**0.500**	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
abalone-20_vs_8-9-10	0.500	0.500	0.500	0.519	**0.526**	0.526	0.507	0.507	0.507
poker-8-9_vs_5	**0.500**	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
abalone-17_vs_7-8-9-10	0.503	0.503	0.503	**0.527**	0.526	0.526	0.523	0.526	0.526

Table 8.20: *Average F1 score scores for* KNN *classifier*

DATASET	hold-out			train			cv		
	precision	recall	balanced	precision	recall	balanced	precision	recall	balanced
adult	0.605	0.609	0.609	0.602	0.620	0.620	0.616	0.630	**0.631**
bank_additional	0.455	0.460	0.455	0.456	0.472	0.463	0.470	**0.476**	0.470
page-blocks0	0.761	0.761	0.761	0.769	0.787	**0.788**	0.759	0.786	0.786
glass1	0.551	0.607	0.594	0.600	**0.657**	0.653	0.505	0.626	0.646
glass0	0.569	0.566	0.568	0.564	**0.670**	0.662	0.575	0.665	0.645
ecoli-0-6-7_vs_5	0.655	0.655	0.655	0.773	**0.777**	0.777	0.755	0.773	0.754
ecoli-0-6-7_vs_3-5	0.639	0.639	0.639	**0.734**	0.730	0.730	0.732	0.693	0.713
ecoli-0-2-6-7_vs_3-5	0.607	0.607	0.607	0.712	0.712	0.712	**0.761**	0.718	0.718
ecoli-0-1_vs_2-3-5	**0.782**	0.782	0.782	0.782	0.782	0.782	0.779	0.772	0.772
haberman	0.336	0.342	0.342	0.283	0.380	0.381	0.240	**0.429**	0.394
ecoli1	0.720	0.717	0.717	0.735	0.751	0.732	0.688	**0.766**	0.761
ecoli2	0.827	0.827	0.827	0.835	0.858	0.851	0.819	**0.868**	0.861
ecoli3	0.476	0.476	0.476	0.493	0.609	0.642	0.544	**0.667**	0.661
ecoli-0-1-4-7_vs_2-3-5-6	0.737	0.737	0.737	0.728	0.780	**0.785**	0.770	0.766	0.769
yeast-1_vs_7	0.165	0.165	0.165	0.343	0.327	0.327	**0.377**	0.350	0.350
page-blocks-1-3_vs_4	0.517	0.525	0.525	**0.779**	0.763	0.763	0.627	0.740	0.699
yeast-2_vs_8	0.444	0.444	0.444	0.552	**0.683**	0.683	0.679	0.679	0.679
yeast-0-3-5-9_vs_7-8	0.328	0.328	0.328	0.264	0.366	0.366	0.387	**0.433**	0.423
yeast-2_vs_4	0.737	0.737	0.737	0.760	0.760	0.760	0.762	**0.768**	0.768
yeast-0-5-6-7-9_vs_4	0.401	0.401	0.401	0.319	0.385	0.385	0.329	**0.460**	0.454
yeast-1-4-5-8_vs_7	0.053	0.053	0.053	0.053	0.053	0.053	0.072	**0.078**	0.078
pima	0.574	0.583	0.583	0.539	0.607	0.592	0.512	**0.647**	0.624
winequality-white-3_vs_7	**0.000**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
yeast-1-2-8-9_vs_7	0.047	0.047	0.047	0.142	**0.152**	0.152	0.130	0.139	0.139
yeast-0-2-5-6_vs_3-7-8-9	0.460	0.501	0.501	0.470	0.584	0.584	0.475	**0.607**	0.607
flare-F	0.140	0.177	0.177	0.083	0.291	0.287	0.295	0.345	**0.345**
kr-vs-k-zero_vs_eight	0.613	0.613	0.613	0.662	0.647	0.647	**0.684**	0.664	0.664
poker-8_vs_6	0.099	0.099	0.099	**0.287**	0.287	0.287	0.202	0.202	0.202
winequality-white-3-9_vs_5	0.000	0.000	0.000	**0.028**	0.028	0.028	0.027	0.027	0.027
yeast3	0.731	0.734	0.734	0.696	0.722	0.722	0.716	0.750	**0.751**
poker-8-9_vs_6	0.079	0.079	0.079	**0.383**	0.383	0.383	0.343	0.343	0.343
yeast6	0.506	0.506	0.506	0.479	0.473	0.480	0.552	**0.569**	0.560
yeast5	0.525	0.534	0.534	0.504	0.608	0.612	0.498	0.652	**0.675**
yeast4	0.177	0.177	0.177	0.253	0.316	0.316	0.290	**0.362**	0.356
winequality-red-4	0.038	0.038	0.038	0.051	0.048	0.048	0.096	**0.098**	0.098
abalone-19_vs_10-11-12-13	**0.000**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
abalone-20_vs_8-9-10	0.000	0.000	0.000	0.065	**0.087**	0.087	0.027	0.027	0.027
poker-8-9_vs_5	**0.000**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
abalone-17_vs_7-8-9-10	0.013	0.013	0.013	**0.094**	0.093	0.093	0.083	0.093	0.093

Table 8.21: *Average Gmean scores for* KNN *classifier*

DATASET	hold-out			train			cv		
	precision	recall	balanced	precision	recall	balanced	precision	recall	balanced
adult	0.715	0.720	0.720	0.710	0.735	0.735	0.722	0.743	**0.743**
bank_additional	0.599	0.606	0.600	0.601	0.620	0.609	0.615	**0.623**	0.615
page-blocks0	0.824	0.825	0.825	0.828	0.865	0.865	0.810	**0.871**	0.871
glass1	0.633	0.686	0.673	0.679	0.726	**0.726**	0.597	0.639	0.719
glass0	0.666	0.657	0.659	0.660	**0.753**	0.743	0.665	0.732	0.732
ecoli-0-6-7_vs_5	0.737	0.737	0.737	0.841	0.851	0.851	0.847	**0.870**	0.868
ecoli-0-6-7_vs_3-5	0.730	0.730	0.730	0.801	0.801	0.801	**0.824**	0.821	0.822
ecoli-0-2-6-7_vs_3-5	0.710	0.710	0.710	0.778	0.778	0.778	0.827	**0.831**	0.826
ecoli-0-1_vs_2-3-5	0.818	0.818	0.818	0.817	0.817	0.817	**0.844**	0.843	0.843
haberman	0.487	0.494	0.494	0.434	0.542	0.539	0.388	0.531	**0.555**
ecoli1	0.818	0.818	0.818	0.799	0.846	0.811	0.742	**0.859**	0.827
ecoli2	0.906	0.906	0.906	0.898	0.921	0.915	0.864	**0.930**	0.918
ecoli3	0.645	0.645	0.645	0.636	0.813	0.799	0.683	**0.879**	0.821
ecoli-0-1-4-7_vs_2-3-5-6	0.805	0.805	0.805	0.784	0.843	0.844	0.848	0.858	**0.859**
yeast-1_vs_7	0.274	0.274	0.274	0.468	0.460	0.460	0.512	**0.542**	0.542
page-blocks-1-3_vs_4	0.650	0.656	0.656	0.864	0.851	0.851	0.740	**0.885**	0.827
yeast-2_vs_8	0.486	0.486	0.486	0.615	**0.738**	0.738	0.737	0.737	0.737
yeast-0-3-5-9_vs_7-8	0.473	0.473	0.473	0.393	0.522	0.522	0.540	**0.663**	0.652
yeast-2_vs_4	0.792	0.792	0.792	0.816	0.816	0.816	0.826	**0.849**	0.849
yeast-0-5-6-7-9_vs_4	0.559	0.559	0.559	0.470	0.545	0.545	0.489	**0.679**	0.652
yeast-1-4-5-8_vs_7	0.098	0.098	0.098	0.114	0.114	0.114	0.160	**0.185**	0.185
pima	0.662	0.670	0.670	0.633	0.691	0.679	0.604	**0.715**	0.705
winequality-white-3_vs_7	**0.000**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
yeast-1-2-8-9_vs_7	0.088	0.088	0.088	0.255	**0.280**	0.280	0.218	0.225	0.225
yeast-0-2-5-6_vs_3-7-8-9	0.588	0.627	0.627	0.586	0.703	0.703	0.594	0.750	**0.751**
flare-F	0.273	0.331	0.331	0.185	0.527	0.507	0.473	**0.658**	0.652
kr-vs-k-zero_vs_eight	0.734	0.734	0.734	0.771	0.758	0.758	0.789	**0.798**	0.798
poker-8_vs_6	0.149	0.149	0.149	**0.389**	0.389	0.389	0.271	0.271	0.271
winequality-white-3-9_vs_5	0.000	0.000	0.000	0.057	0.057	0.057	**0.058**	0.058	0.058
yeast3	0.804	0.808	0.808	0.784	0.815	0.815	0.787	**0.854**	0.853
poker-8-9_vs_6	0.135	0.135	0.135	**0.486**	0.486	0.486	0.460	0.460	0.460
yeast6	0.678	0.678	0.678	0.639	0.653	0.653	0.724	**0.803**	0.778
yeast5	0.671	0.682	0.682	0.637	0.788	0.775	0.621	**0.893**	0.853
yeast4	0.317	0.317	0.317	0.412	0.488	0.488	0.448	**0.617**	0.599
winequality-red-4	0.105	0.105	0.105	0.121	0.121	0.121	0.234	**0.240**	0.240
abalone-19_vs_10-11-12-13	**0.000**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
abalone-20_vs_8-9-10	0.000	0.000	0.000	0.139	**0.178**	0.178	0.055	0.055	0.055
poker-8-9_vs_5	**0.000**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
abalone-17_vs_7-8-9-10	0.037	0.037	0.037	**0.211**	0.211	0.211	0.193	0.206	0.206

D Additional results for surrogate criteria for gradient optimisation

Table 8.22: *Precision measure values of all compared methods*

DATASET	COSMOS MIN	COSMOS BAL	COSMOS MAX	focal loss 0	focall loss 2	bce
adult	**0.89**	0.553	0.383	0.683	0.699	0.566
page-blocks0	0.796	0.644	0.374	0.861	**0.869**	0.713
bank_additional	**0.65**	0.428	0.331	0.561	0.587	0.463
MiniBooNE_PID	**0.995**	0.974	0.869	0.956	0.95	0.931
ecoli1	**0.802**	0.675	0.584	0.802	0.785	0.719
ecoli3	0.603	0.483	0.415	**0.647**	0.633	0.53
glass0	**0.717**	0.618	0.495	0.700	0.697	0.645
glass1	**0.698**	0.593	0.467	0.637	0.651	0.621
haberman	**0.577**	0.458	0.265	0.447	0.450	0.411
pima	**0.774**	0.613	0.432	0.633	0.609	0.603
yeast-0-2-5-6_vs_3-7-8-9	**0.730**	0.428	0.151	0.692	0.673	0.451
yeast-0-3-5-9_vs_7-8	**0.577**	0.245	0.141	0.567	0.489	0.278
yeast-0-5-6-7-9_vs_4	**0.641**	0.375	0.220	0.551	0.576	0.439
yeast3	0.777	0.632	0.482	**0.782**	0.753	0.655
yeast4	0.180	0.130	0.092	0.462	**0.489**	0.309

Table 8.23: *Recall measure values of all compared methods*

DATASET	COSMOS MIN	COSMOS BAL	COSMOS MAX	focal loss 0	focall loss 2	bce
adult	0.309	0.856	**0.980**	0.615	0.605	0.810
page-blocks0	0.831	0.932	**0.985**	0.815	0.800	0.916
bank_additional	0.406	0.901	**0.971**	0.516	0.519	0.806
MiniBooNE_PID	0.736	0.892	**0.979**	0.952	0.951	0.970
ecoli1	0.735	0.888	**0.967**	0.780	0.751	0.858
ecoli3	0.789	0.880	**0.931**	0.601	0.601	0.829
glass0	0.340	0.851	**0.954**	0.706	0.717	0.806
glass1	0.229	0.745	**0.945**	0.624	0.621	0.700
haberman	0.215	0.596	**0.995**	0.284	0.303	0.504
pima	0.299	0.725	**0.969**	0.606	0.587	0.693
yeast-0-2-5-6_vs_3-7-8-9	0.451	0.667	**0.881**	0.477	0.461	0.617
yeast-0-3-5-9_vs_7-8	0.264	0.588	**0.808**	0.260	0.268	0.480
yeast-0-5-6-7-9_vs_4	0.350	0.698	**0.859**	0.349	0.369	0.592
yeast3	0.758	0.866	**0.919**	0.726	0.733	0.827
yeast4	0.756	0.843	**0.855**	0.262	0.290	0.525

Table 8.24: BAC *measure values of all compared methods*

DATASET	COSMOS MIN	COSMOS BAL	COSMOS MAX	focal loss 0	focall loss 2	bce
adult	0.648	**0.818**	0.739	0.762	0.761	0.806
page-blocks0	0.904	0.937	0.896	0.900	0.893	**0.937**
bank_additional	0.689	**0.874**	0.861	0.732	0.736	0.844
MiniBooNE_PID	0.863	0.915	0.799	**0.919**	0.910	0.893
ecoli1	0.840	**0.880**	0.880	0.860	0.843	0.878
ecoli3	0.864	0.885	**0.888**	0.781	0.779	0.872
glass0	0.633	**0.795**	0.738	0.773	0.779	0.792
glass1	0.585	0.727	0.675	0.713	0.716	**0.730**
haberman	0.579	**0.668**	0.501	0.578	0.586	0.621
pima	0.626	**0.739**	0.642	0.708	0.692	0.723
yeast-0-2-5-6_vs_3-7-8-9	0.716	**0.784**	0.668	0.726	0.718	0.766
yeast-0-3-5-9_vs_7-8	0.621	**0.694**	0.632	0.618	0.618	0.670
yeast-0-5-6-7-9_vs_4	0.664	**0.784**	0.756	0.659	0.669	0.754
yeast3	0.866	**0.902**	0.896	0.851	0.851	0.886
yeast4	0.816	**0.821**	0.777	0.625	0.639	0.740

Table 8.25: *F1 score values of all compared methods*

DATASET	COSMOS MIN	COSMOS BAL	COSMOS MAX	focal loss 0	focall loss 2	bce
adult	0.456	**0.672**	0.551	0.645	0.648	0.666
page-blocks0	0.813	0.761	0.541	**0.837**	0.832	0.801
bank_additional	0.497	0.580	0.493	0.537	0.549	**0.588**
MiniBooNE_PID	0.846	0.931	0.920	**0.954**	0.950	0.950
ecoli1	0.765	0.766	0.726	**0.786**	0.764	0.780
ecoli3	**0.683**	0.623	0.574	0.620	0.613	0.645
glass0	0.444	**0.713**	0.650	0.692	0.701	0.711
glass1	0.339	0.657	0.625	0.628	0.633	**0.657**
haberman	0.310	**0.512**	0.419	0.345	0.359	0.449
pima	0.429	**0.663**	0.597	0.618	0.597	0.644
yeast-0-2-5-6_vs_3-7-8-9	0.544	0.518	0.257	**0.558**	0.543	0.516
yeast-0-3-5-9_vs_7-8	**0.358**	0.345	0.240	0.349	0.342	0.349
yeast-0-5-6-7-9_vs_4	0.444	0.483	0.345	0.421	0.443	**0.495**
yeast3	**0.766**	0.729	0.629	0.752	0.742	0.729
yeast4	0.29	0.226	0.167	0.326	0.353	**0.381**

Table 8.26: *Gmean measure values of all compared methods*

DATASET	COSMOS MIN	COSMOS BAL	COSMOS MAX	focal loss 0	focall loss 2	bce
adult	0.551	**0.817**	0.699	0.747	0.745	0.806
page-blocks0	0.901	**0.936**	0.892	0.896	0.888	0.936
bank_additional	0.626	**0.873**	0.853	0.699	0.702	0.843
MiniBooNE_PID	0.854	0.915	0.779	**0.919**	0.909	0.889
ecoli1	0.833	**0.880**	0.875	0.855	0.837	0.877
ecoli3	0.860	0.884	**0.887**	0.757	0.757	0.869
glass0	0.550	**0.791**	0.704	0.765	0.773	0.788
glass1	0.460	0.724	0.616	0.706	0.708	**0.729**
haberman	0.446	**0.659**	0.044	0.495	0.508	0.606
pima	0.532	**0.738**	0.550	0.700	0.683	0.722
yeast-0-2-5-6_vs_3-7-8-9	0.658	**0.774**	0.632	0.679	0.668	0.749
yeast-0-3-5-9_vs_7-8	0.504	**0.685**	0.602	0.499	0.503	0.640
yeast-0-5-6-7-9_vs_4	0.580	**0.778**	0.743	0.578	0.593	0.732
yeast3	0.859	**0.901**	0.895	0.841	0.843	0.884
yeast4	0.813	**0.820**	0.772	0.504	0.531	0.705

Bibliography

[1] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[2] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):429–449, 2002.

[3] Mohammed Khalilia, Sounak Chakraborty, and Mihail Popescu. Predicting disease risks from highly imbalanced data using random forest. *BMC Medical Informatics and Decision Making*, 11(1):1–13, 2011.

[4] Kang Fu, Dawei Cheng, Yi Tu, and Liqing Zhang. Credit card fraud detection using convolutional neural networks. In *International Conference on Neural Information Processing*, pages 483–490. Springer, 2016.

[5] David A Cieslak, Nitesh V Chawla, and Aaron Striegel. Combating imbalance in network intrusion datasets. In *GrC*, pages 732–737, 2006.

[6] Dariusz Brzezinski, Jerzy Stefanowski, Robert Susmaga, and Izabela Szczech. On the dynamics of classification measures for imbalanced and streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, 31(8):2868–2878, 2019.

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

[8] Ludmila I Kuncheva. *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2014.

[9] Ryszard Stanislaw Michalski, Jaime Guillermo Carbonell, and Tom M Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[10] Taiwo Oladipupo Ayodele. Types of machine learning algorithms. *New Advances in Machine Learning*, 3(19-48):5–1, 2010.

[11] Mikel Galar, Alberto Fernández, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition*, 44(8):1761–1776, 2011.

[12] Masashi Sugiyama, Han Bao, Takashi Ishida, Nan Lu, and Tomoya Sakai. *Machine learning from weak supervision: An empirical risk minimization approach*. MIT Press, 2022.

[13] Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

[14] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM Computing Surveys (CSUR)*, 31(3):264–323, 1999.

[15] Jasmine Irani, Nitin Pise, and Madhura Phatak. Clustering techniques and the similarity measures used in clustering: A survey. *International journal of computer applications*, 134(7):9–14, 2016.

[16] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *Journal of Machine Learning Research*, 22(164):1–20, 2021.

[17] Michal Wozniak. *Hybrid classifiers: methods of data, knowledge, and classifier combination*, volume 519. Springer, 2013.

[18] Sotiris B Kotsiantis, Dimitris Kanellopoulos, and Panagiotis E Pintelas. Data preprocessing for supervised leaning. *International Journal of Computer Science*, 1(2):111–117, 2006.

[19] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C Prati, Bartosz Krawczyk, Francisco Herrera, Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C Prati, et al. Cost-sensitive learning. *Learning from imbalanced data sets*, pages 63–78, 2018.

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

[21] Nathalie Japkowicz and Zois Boukouvalas. *Machine learning evaluation: towards reliable and responsible AI*. Cambridge University Press, 2024.

[22] Tzu-Tsung Wong. Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. *Pattern Recognition*, 48(9):2839–2846, 2015.

[23] Thomas G Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

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

[25] Nathalie Japkowicz. Why question machine learning evaluation methods. In *AAAI workshop on evaluation methods for machine learning*, volume 6. University of Ottawa, 2006.

[26] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. The balanced accuracy and its posterior distribution. In *2010 20th International Conference on Pattern Recognition*, pages 3121–3124. IEEE, 2010.

[27] Miroslav Kubat, Robert C Holte, and Stan Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30:195–215, 1998.

[28] Nathalie Japkowicz. Assessment metrics for imbalanced learning. *Imbalanced Learning: Foundations, Algorithms, and Applications*, pages 187–206, 2013.

[29] David Hand and Peter Christen. A note on using the F-measure for evaluating record linkage algorithms. *Statistics and Computing*, 28(3):539–547, 2018.

[30] David J Hand and Christoforos Anagnostopoulos. When is the area under the receiver operating characteristic curve an appropriate measure of classifier performance? *Pattern Recognition Letters*, 34(5):492–495, 2013.

[31] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[32] David J Hand and Robert J Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45:171–186, 2001.

[33] Cesar Ferri, Peter Flach, José Hernández-Orallo, and Athmane Senad. Modifying ROC curves to incorporate predicted probabilities. In *Proceedings of the Second Workshop on ROC Analysis in Machine Learning*, volume 4140, pages 33–40. International Conference on Machine Learning, 2005.

[34] Vicente García, Ramon A Mollineda, and J Salvador Sánchez. Theoretical analysis of a performance measure for imbalanced data. In *2010 20th International Conference on Pattern Recognition*, pages 617–620. IEEE, 2010.

[35] Sabri Boughorbel, Fethi Jarray, and Mohammed El-Anbari. Optimal classifier for imbalanced data using matthews correlation coefficient metric. *PloS One*, 12(6):e0177678, 2017.

[36] Ayfer Ezgi Yilmaz and Haydar Demirhan. Weighted kappa measures for ordinal multi-class classification performance. *Applied Soft Computing*, 134:110020, 2023.

[37] César Ferri, José Hernández-Orallo, and R Modroiu. An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27–38, 2009.

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

[39] Claude Nadeau and Yoshua Bengio. Inference for the generalization error. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[40] Ethem Alpaydm. Combined 5× 2 cv f test for comparing supervised classification learning algorithms. *Neural Computation*, 11(8):1885–1892, 1999.

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

[42] Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in Statistics: Methodology and Distribution*, pages 196–202. Springer, 1992.

[43] Steven L Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1:317–328, 1997.

[44] Ronald Aylmer Fisher. *Statistical methods and scientific inference (2nd edition)*. Hafner Publishing Co., 1959.

[45] John W Tukey. Comparing individual means in the analysis of variance. *Biometrics*, pages 99–114, 1949.

[46] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[47] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *Journal of Machine Learning Research*, 18(77):1–36, 2017.

[48] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[49] Sotiris B Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, 39:261–283, 2013.

[50] Lior Rokach and Oded Maimon. Decision trees. *Data Mining and Knowledge Discovery Handbook*, pages 165–192, 2005.

[51] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

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

[53] Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[54] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[55] Bayya Yegnanarayana. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[56] Marius-Constantin Popescu, Valentina E Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8(7):579–588, 2009.

[57] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):6999–7019, 2021.

[58] Anthony Gillioz, Jacky Casas, Elena Mugellini, and Omar Abou Khaled. Overview of the transformer-based models for nlp tasks. In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*, pages 179–183. IEEE, 2020.

[59] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

[60] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[61] Francis Galton. Vox populi. *Nature*, 75(1949):450–451, Mar 1907.

[62] Marie Jean Caritat. *Essai sur l'application de l'analyse 'a la probabilit'e des d'ecisions rendues 'a la pluralit'e des voix*. De l'Imprimerie Royale, 1785.

[63] Stephen B Vardeman and Max D Morris. Majority voting by independent classifiers can increase error rates. *The American Statistician*, 67(2):94–96, 2013.

[64] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[65] Padraig Cunningham and John Carney. Diversity versus quality in classification ensembles based on feature selection. In *European Conference on Machine Learning*, pages 109–116. Springer, 2000.

[66] Michał Żak and Michał Woźniak. Performance analysis of binarization strategies for multi-class imbalanced data classification. In *International Conference on Computational Science*, pages 141–155. Springer, 2020.

[67] Anne MP Canuto, Marjory CC Abreu, Lucas de Melo Oliveira, Joao C Xavier Jr, and Araken de M Santos. Investigating the influence of the choice of the ensemble members in accuracy and diversity of selection-based and fusion-based methods for ensembles. *Pattern Recognition Letters*, 28(4):472–486, 2007.

[68] George DC Cavalcanti, Luiz S Oliveira, Thiago JM Moura, and Guilherme V Carvalho. Combining diversity measures for ensemble pruning. *Pattern Recognition Letters*, 74:38–45, 2016.

[69] Ludmila I Kuncheva and Juan J Rodríguez. A weighted voting framework for classifiers ensembles. *Knowledge and Information Systems*, 38:259–275, 2014.

[70] Seng Zian, Sameem Abdul Kareem, and Kasturi Dewi Varathan. An empirical evaluation of stacked ensembles with different meta-learners in imbalanced classification. *IEEE Access*, 9:87434–87452, 2021.

[71] Merijn Van Erp, Louis Vuurpijl, and Lambert Schomaker. An overview and comparison of voting methods for pattern recognition. In *Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition*, pages 195–200. IEEE, 2002.

[72] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[73] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[74] Adele Cutler, D Richard Cutler, and John R Stevens. Random forests. *Ensemble Machine Learning: Methods and Applications*, pages 157–175, 2012.

[75] Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2:165–193, 2015.

[76] J Macqueen. *Some methods for classification and analysis of multivariate observations*. University of California Press, 1967.

[77] Bartosz Krawczyk and Bogusław Cyganek. Selecting locally specialised classifiers for one-class classification ensembles. *Pattern Analysis and Applications*, 20:427–439, 2017.

[78] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD'96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, volume 96, pages 226–231, 1996.

[79] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. *ACM Sigmod Record*, 28(2):49–60, 1999.

[80] Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[81] 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.

[82] Alberto Fernández, Salvador Garcia, Francisco Herrera, and Nitesh V Chawla. Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of Artificial Intelligence Research*, 61:863–905, 2018.

[83] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International Conference on Intelligent Computing*, pages 878–887. Springer, 2005.

[84] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks*, pages 1322–1328. IEEE, 2008.

[85] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1):20–29, 2004.

[86] Jaedong Lee, Noo-ri Kim, and Jee-Hyong Lee. An over-sampling technique with rejection for imbalanced class learning. In *Proceedings of the 9th International*

Conference on Ubiquitous Information Management and Communication, pages 1–6, 2015.

[87] S. Gazzah and N. E. B. Amara. New oversampling approaches based on polynomial fitting for imbalanced data sets. In *2008 The Eighth IAPR International Workshop on Document Analysis Systems*, pages 677–684, Sept 2008.

[88] Sebastián Maldonado, Carla Vairetti, Alberto Fernandez, and Francisco Herrera. Fw-smote: A feature-weighted oversampling approach for imbalanced classification. *Pattern Recognition*, 124:108511, 2022.

[89] Georgios Douzas, Fernando Bacao, and Felix Last. Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *Information Sciences*, 465:1–20, 2018.

[90] Zhaozhao Xu, Derong Shen, Tiezheng Nie, Yue Kou, Nan Yin, and Xi Han. A cluster-based oversampling algorithm combining smote and k-means for imbalanced medical data. *Information Sciences*, 572:574–589, 2021.

[91] Damien Dablain, Bartosz Krawczyk, and Nitesh V Chawla. Deepsmote: Fusing deep learning and smote for imbalanced data. *IEEE Transactions on Neural Networks and Learning Systems*, 34(9):6390–6404, 2022.

[92] Peter Hart. The condensed nearest neighbor rule (corresp.). *IEEE Transactions on Information Theory*, 14(3):515–516, 1968.

[93] Dennis L Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (3):408–421, 1972.

[94] Ivan Tomek. Two modifications of cnn. *IEEE Transactions on Systems, Man, and Cybernetics*, 1976.

[95] Wei-Chao Lin, Chih-Fong Tsai, Ya-Han Hu, and Jing-Shang Jhang. Clustering-based undersampling in class-imbalanced data. *Information Sciences*, 409:17–26, 2017.

[96] Michał Koziarski. Radial-based undersampling for imbalanced data classification. *Pattern Recognition*, 102:107262, 2020.

[97] Salvador García and Francisco Herrera. Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evolutionary Computation*, 17(3):275–306, 2009.

[98] 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):727–736, 2017.

[99] Michał Koziarski, Colin Bellinger, and Michał Woźniak. RB-CCR: radial-based combined cleaning and resampling algorithm for imbalanced data classification. *Machine Learning*, 110:3059–3093, 2021.

[100] Qiang Wang. A hybrid sampling svm approach to imbalanced data classification. In *Abstract and Applied Analysis*, volume 2014, page 972786. Wiley Online Library, 2014.

[101] Abdul Sattar Palli, Jafreezal Jaafar, Manzoor Ahmed Hashmani, Heitor Murilo Gomes, and Abdul Rehman Gilal. A hybrid sampling approach for imbalanced binary and multi-class data using clustering analysis. *IEEE Access*, 10:118639–118653, 2022.

[102] Charles X Ling and Victor S Sheng. Cost-sensitive learning and the class imbalance problem. *Encyclopedia of Machine Learning*, 2011:231–235, 2008.

[103] Harshit Dubey and Vikram Pudi. Class based weighted k-nearest neighbor over imbalance dataset. In *Advances in Knowledge Discovery and Data Mining: 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings, Part II 17*, pages 305–316. Springer, 2013.

[104] Konstantinos Veropoulos, Colin Campbell, Nello Cristianini, et al. Controlling the sensitivity of support vector machines. In *Proceedings of the International Joint Conference on AI*, volume 55, page 60. Stockholm, 1999.

[105] Stefan Lessmann. Solving imbalanced classification problems with support vector machines. In *IC-AI*, volume 4, pages 214–220, 2004.

[106] Liangxiao Jiang, Lungan Zhang, Liangjun Yu, and Dianhong Wang. Class-specific attribute weighted naive Bayes. *Pattern Recognition*, 88:321–330, 2019.

[107] Justin M Johnson and Taghi M Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of big data*, 6(1):1–54, 2019.

[108] Shoujin Wang, Wei Liu, Jia Wu, Longbing Cao, Qinxue Meng, and Paul J Kennedy. Training deep neural networks on imbalanced data sets. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 4368–4374. IEEE, 2016.

[109] Yuri Sousa Aurelio, Gustavo Matheus De Almeida, Cristiano Leite de Castro, and Antonio Padua Braga. Learning from imbalanced data sets with weighted cross-entropy function. *Neural Processing Letters*, 50:1937–1949, 2019.

[110] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[111] 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, 2011.

[112] Cen Li. Classifying imbalanced data using a bagging ensemble variation (BEV). In *Proceedings of the 45th Annual Southeast Regional Conference*, pages 203–208, 2007.

[113] Shohei Hido, Hisashi Kashima, and Yutaka Takahashi. Roughly balanced bagging for imbalanced data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 2(5-6):412–426, 2009.

[114] Jerzy Błaszczyński and Jerzy Stefanowski. Neighbourhood sampling in bagging for imbalanced data. *Neurocomputing*, 150:529–542, 2015.

[115] Shuo Wang and Xin Yao. Diversity analysis on imbalanced data sets by using ensemble models. In *2009 IEEE Symposium on Computational Intelligence and Data Mining*, pages 324–331. IEEE, 2009.

[116] Zhi Chen, Jiang Duan, Li Kang, and Guoping Qiu. A hybrid data-level ensemble to enable learning from highly imbalanced dataset. *Information Sciences*, 554:157–176, 2021.

[117] Nitesh V Chawla, Aleksandar Lazarevic, Lawrence O Hall, and Kevin W Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In *Knowledge Discovery in Databases: PKDD 2003: 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22-26, 2003. Proceedings 7*, pages 107–119. Springer, 2003.

[118] Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. RUSBoost: Improving classification performance when training data is skewed. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[119] Mikel Galar, Alberto Fernández, Edurne Barrenechea, and Francisco Herrera. EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. *Pattern Recognition*, 46(12):3460–3471, 2013.

[120] Xinmin Tao, Qing Li, Wenjie Guo, Chao Ren, Chenxi Li, Rui Liu, and Junrong Zou. Self-adaptive cost weights-based support vector machine cost-sensitive ensemble for imbalanced data classification. *Information Sciences*, 487:31–56, 2019.

[121] Zhongbin Sun, Qinbao Song, Xiaoyan Zhu, Heli Sun, Baowen Xu, and Yuming Zhou. A novel ensemble method for classifying imbalanced data. *Pattern Recognition*, 48(5):1623–1637, 2015.

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

[123] Wirot Yotsawat, Pakaket Wattuya, and Anongnart Srivihok. A novel method for credit scoring based on cost-sensitive neural network ensemble. *IEEE Access*, 9:78521–78537, 2021.

[124] Jakub Klikowski and Michał Woźniak. Multi sampling random subspace ensemble for imbalanced data stream classification. In *Progress in Computer Recognition Systems 11*, pages 360–369. Springer, 2020.

[125] Sheng Chen and Haibo He. Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach. *Evolving Systems*, 2(1):35–50, 2011.

[126] Abhijeet Godase and Vahida Attar. Classifier ensemble for imbalanced data stream classification. In *Proceedings of the CUBE International Information Technology Conference*, CUBE '12, page 284–289, New York, NY, USA, 2012. Association for Computing Machinery.

[127] Joanna Grzyb, Jakub Klikowski, and Michał Woźniak. Hellinger distance weighted ensemble for imbalanced data stream classification. *Journal of Computational Science*, 51:101314, 2021.

[128] Toshitaka Hayashi and Hamido Fujita. One-class ensemble classifier for data imbalance problems. *Applied Intelligence*, 52(15):17073–17089, 2022.

[129] Zhongliang Zhang, Bartosz Krawczyk, Salvador Garcia, Alejandro Rosales-Pérez, and Francisco Herrera. Empowering one-vs-one decomposition with ensemble learning for multi-class imbalanced data. *Knowledge-Based Systems*, 106:251–263, 2016.

[130] Michael TM Emmerich and André H Deutz. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural Computing*, 17:585–609, 2018.

[131] Jin-Hee Cho, Yating Wang, Ray Chen, Kevin S Chan, and Ananthram Swami. A survey on modeling and optimizing multi-objective systems. *IEEE Communications Surveys & Tutorials*, 19(3):1867–1901, 2017.

[132] Vilfredo Pareto. *Cours d'économie politique*, volume 1. Librairie Droz, 1964.

[133] Antonio López Jaimes, Saúl Zapotecas Martınez, Carlos A Coello Coello, et al. An introduction to multiobjective optimization techniques. *Optimization in Polymer Processing*, 1:29, 2009.

[134] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[135] Robin C Purshouse and Peter J Fleming. Why use elitism and sharing in a multi-objective genetic algorithm? In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary computation*, pages 520–527, 2002.

[136] Shanu Verma, Millie Pant, and Vaclav Snasel. A comprehensive review on NSGA-II for multi-objective combinatorial optimization problems. *IEEE Access*, 9:57757–57791, 2021.

[137] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. *TIK Report*, 103, 2001.

[138] Nicola Beume, Boris Naujoks, and Michael Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[139] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2013.

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

[141] Chigozie Enyinna Nwankpa. Advances in optimisation algorithms and techniques for deep learning. *Advances in Science, Technology and Engineering Systems Journal*, 5(5):563–577, 2020.

[142] Jean-Antoine Désidéri. Multiple-gradient descent algorithm (MGDA) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318, 2012.

[143] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[144] Michael Ruchte and Josif Grabocka. Scalable pareto front approximation for deep multi-objective learning. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 1306–1311. IEEE, 2021.

[145] 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.

[146] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[147] José Figueira, Salvatore Greco, and Matthias Ehrgott. *Multiple criteria decision analysis: state of the art surveys.* Springer, 2016.

[148] Evangelos Triantaphyllou. *Multi-Criteria Decision Making Methods: A Comparative Study.* Springer, 2000.

[149] Hamed Taherdoost and Mitra Madanchian. A comprehensive overview of the electre method in multi-criteria decision-making. *Journal of Management Science Engineering Research*, pages 5–16, 2023.

[150] Majid Behzadian, Reza Baradaran Kazemzadeh, Amir Albadvi, and Mohammad Aghdasi. PROMETHEE: A comprehensive literature review on methodologies and applications. *European Journal of Operational Research*, 200(1):198–215, 2010.

[151] Eyke Hüllermeier and Roman Słowiński. Preference learning and multiple criteria decision aiding: differences, commonalities, and synergies–part i. *4OR*, 22:179–209, 2024.

[152] 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.

[153] Everlandio R. Q. Fernandes, Andre C. P. L. F. de Carvalho, and Xin Yao. Ensemble of classifiers based on multiobjective genetic sampling for imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 32(6):1104–1115, 2020.

[154] Seyed Ehsan Roshan and Shahrokh Asadi. Improvement of bagging performance for classification of imbalanced datasets using evolutionary multi-objective optimization. *Engineering Applications of Artificial Intelligence*, 87:103319, 2020.

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

[156] Alberto Fernández, Cristobal José Carmona, Maria Jose 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.

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

[158] 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.

[159] Yi Liu, Yanzhen Wang, Xiaoguang Ren, Hao Zhou, and Xingchun Diao. A classification method based on feature selection for imbalanced data. *IEEE Access*, 7:81794–81807, 2019.

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

[161] Shuo Wang, Leandro L Minku, and Xin Yao. A multi-objective ensemble method for online class imbalance learning. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 3311–3318. IEEE, 2014.

[162] Anju Jain, Saroj Ratnoo, and Dinesh Kumar. A novel multi-objective genetic algorithm approach to address class imbalance for disease diagnosis. *International Journal of Information Technology*, pages 1–16, 2020.

[163] Szymon Wojciechowski. Multi-objective evolutionary undersampling algorithm for imbalanced data classification. In *International Conference on Computational Science*, pages 118–127. Springer, 2021.

[164] Marzieh Hajizadeh Tahan and Shahrokh Asadi. EMDID: Evolutionary multi-objective discretization for imbalanced datasets. *Information Sciences*, 432:442–461, 2018.

[165] Ayşegül Aşkan and Serpil Sayın. SVM classification for imbalanced data sets using a multiobjective optimization framework. *Annals of Operations Research*, 216:191–203, 2014.

[166] Shounak Datta and Swagatam Das. Multiobjective support vector machines: Handling class imbalance with pareto optimality. *IEEE Transactions on Neural Networks and Learning Systems*, 30(5):1602–1608, 2019.

[167] Yanjiao Li, Jie Zhang, Sen Zhang, Wendong Xiao, and Zhiqiang Zhang. Multi-objective optimization-based adaptive class-specific cost extreme learning machine for imbalanced classification. *Neurocomputing*, 496:107–120, 2022.

[168] Alex Shenfield and Shahin Rostami. Multi-objective evolution of artificial neural networks in multi-class medical diagnosis problems with class imbalance. In *2017 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–8, 2017.

[169] Nicolò Felicioni et al. Multi-objective blended ensemble for highly imbalanced sequence aware tweet engagement prediction. In *Proceedings of the Recommender Systems Challenge 2020*, pages 29–33. 2020.

[170] Hardik H. Maheta and Vipul K. Dabhi. Classification of imbalanced data sets using multi objective genetic programming. In *2015 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–6, 2015.

[171] Marwa Chabbouh, Slim Bechikh, Chih-Cheng Hung, and Lamjed Ben Said. Multi-objective evolution of oblique decision trees for imbalanced data binary classification. *Swarm and Evolutionary Computation*, 49:1–22, 2019.

[172] Pietro Ducange, Beatrice Lazzerini, and Francesco Marcelloni. Multi-objective genetic fuzzy classifiers for imbalanced and cost-sensitive datasets. *Soft Computing*, 14:713–728, 2010.

[173] 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, 2013.

[174] 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.

[175] Jesús Alcalá-Fdez et al. KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17, 2011.

[176] Dheeru Dua and Casey Graff. UCI Machine Learning Repository, 2017.

[177] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[178] J. Blank and K. Deb. pymoo: Multi-Objective Optimization in Python. *IEEE Access*, 8:89497–89509, 2020.

[179] Katarzyna Stapor et al. How to design the fair experimental classifier evaluation. *Applied Soft Computing*, 104:107219, 2021.

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

[181] Wuxing Chen, Kaixiang Yang, Zhiwen Yu, Yifan Shi, and CL Philip Chen. A survey on imbalanced learning: latest research, applications and future directions. *Artificial Intelligence Review*, 57(6):137, 2024.

[182] Guillaume Lemaître et al. 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.

[183] György Kovács. smote-variants: a Python Implementation of 85 Minority Oversampling Techniques. *Neurocomputing*, 366:352–354, 2019.

[184] Weronika Węgier, Michał Koziarski, and Michał Woźniak. Optimized hybrid imbalanced data sampling for decision tree training. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pages 339–342, 2023.

[185] 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.

[186] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

[187] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[188] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[189] Przemyslaw Biecek and Tomasz Burzykowski. *Explanatory model analysis: explore, explain, and examine predictive models*. Chapman and Hall/CRC, 2021.

[190] Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.