WROCLAW UNIVERSITY OF SCIENCE AND TECHNOLOGY

Ensemble methods for imbalanced data stream classification

by

Jakub Klikowski

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

December 2021

Don't regret, never regret
that you could have done something
in your life and you didn't.
You didn't, because you couldn't.

STANISŁAW LEM

Acknowledgements

First of all, I would like to thank my supervisor, Prof. Michał Woźniak, for all the help in my work, sharing his knowledge during conversations, constantly infecting me with peace and positive thinking, motivating me to action, and for all the answers from my question lists.

I would also like to thank my parents and my brother for their support in every situation, their help in many dilemmas, patience to me and honestly listening to my stories about "machine learning", which did not have to be fully understood.

I thank my beloved girlfriend Asia for her spiritual and substantive help at work on this thesis. I also thank her for all the moments spent together that allowed me to break away from work, calm my mind and really rest.

Thank you very much to all my colleagues from the Department of Computer Systems and Networks, who never refused to help and allowed me to work in a very pleasant atmosphere.

I would like to express my special thanks to Ph.D. Paweł Ksieniewicz for all the help I have received so far, who helped me most at the end of my Master's studies, indicating which path I should go.

Special thanks are also due to Paweł Zyblewski, who accompanied me in many moments, experiencing similar situations, and not once did he help me with the right advice or remark.

At the very end, I would like to thank my cat "Rysiek" for saving me from working alone by spending hours together at my desk (on the desk) and effectively forcing me to take more or less breaks from work by sitting on the computer keyboard and my dog "Luna", whom many times took me out for a walk to air my head and get fresh thoughts.

W pierwszej kolejności chciałbym ogromnie podziękować mojemu promotorowi prof. Michałowi Woźniakowi za szansę jaką dostałem, wszelką pomoc w pracy, dzielenie się swoją wiedzą podczas rozmów, nieustanne zarażaniem spokojem i pozytywnym podejściem do życia, motywowanie mnie do działania oraz za wszystkie odpowiedzi z moich list pytań.

Chciałbym również podziękować moim rodzicom oraz mojemu bratu za wsparcie w każdej sytuacji, pomoc w wielu dylemetach, cierpliwość do mojej osoby oraz szczere wysłuchiwanie moich opowieści na tematy związanie z "uczeniem maszynowym", które nie do końca musiały być w pełni zrozumiałe.

Dziękuję mojej ukochanej dziewczynie Asi za pomoc duchową i merytoryczną podczas pisania tej rozprawy i w wielu innych sytuacjach. Dziękuję również, za wszystkie chwile spędzone razem, które pozwalały mi się oderwać od pracy, uspokoić myśli i naprawdę odpocząć.

Bardzo dziękuje kolegom i koleżankom z Katedry Systemów i Sieci Komputerowych, którzy nigdy nie odmówili pomocy i pozwolili na współpracę w bardzo przyjemnej atmosferze.

Szczególne podziekowania chciałbym skierować do Dr. Pawła Ksieniewicza za wszelką pomoc, który najbardziej mi pomógł pod koniec studiów magisterskich, między innymi wskazując, jaką ścieżką podążać dalej.

Specjalne podziękowanie należy się również Pawłowi Zyblewskiemu, już ze stopniem doktora, który towarzyszył mi w wielu momentach, doświadczając podobnych sytuacji i nie jednokrotnie ratował mnie trafnymi radami lub uwagami.

Na sam koniec chciałbym podziękować mojemu kotu "Ryśkowi" za ratowanie mnie od samotnej pracy przez wspólnie spędzone godziny przy biurku (na biurku) oraz skuteczne zmuszanie mnie do robienia sobie większych lub mniejszych przerw od pracy poprzez siadanie na klawiaturze komputera oraz pieskowi "Lunie", która nie raz wyciągnęła mnie na spacer, żeby przewietrzyć głowę i nabrać świeżych myśli.

Abbreviations

ADS	accumulated data storage
CA	clustering algorithm
CM	clusters consistency metric
BAC	balanced accuracy
BCL	base classifier learning method
BP	balance parameter
DTC	decision tree classifier
FN	false negative - number of wrong classified positive examples
FNR	false negative rate
FP	false positive - number of wrong classified negative examples
FPR	false positive rate
GNB	Gaussian naive Bayes classifier
IR	imbalance ratio
KNN	k nearest neighbours classifier
MPL	maximum performance loss
MLP	multi-layer perceptron
OSM	oversampling method
$OCSVM$	one class support vector machine classifier
PPM	predictive performance measure
RT	restoration time

SVM	support vector machine classifier
TN	true negative - number of correct classified negative examples
TNR	true negative rate (specificity)
TP	true positive - number of correct classified positive examples
TPR	true positive rate (recall, sensitivity)
USM	undersampling method

Symbols

$A_{t,i}^{m}$	age parameter of $\Psi_{t,i}^{m}$
$A_{t,j}^{M}$	age parameter of $\Psi_{t,j}^{M}$
α_i	dual coefficients parameter
B_t	bootstraped samples from t-th data chunk
β	$Fscore$ metric parameter
c_p	number of centroid pairs
C	slack variable strength parameter
$C_{t,k}^{m}$	clusters of the t-th minority data chunk
$C_{t,k}^{M}$	clusters of the t-th majority data chunk
$CL_{t,i}^{m}$	"competence" level parameter of $\Psi_{t,i}^{m}$
$CL_{t,j}^{M}$	"competence" level parameter of $\Psi_{t,j}^{M}$
d, d'	feature vector dimension
d_h	weight heuristic decay parameter
DS	data stream
DS_t	t-th chunk of data stream DS
DS_t^m	minority data in the t-th chunk
DS_t^M	majority data in the t-th chunk
DSU_t	undersampled t-th data chunk
f	probability density function

$f_{t,i}^m$	decision function of $\Psi_{t,i}^m$
$f_{t,j}^M$	decision function of $\Psi_{t,j}^M$
F	support function
F_i	support function for class i
F_i^k	support function of the k-th classifier for class i
γ	weight degradation parameter
$Gmean$	geometric mean of precision and recall
$Gmean_s$	geometric mean of specificity and sensitivity
h	number of hidden layers
$IF_{t,i}^m$	imbalance ratio parameter of $\Psi_{t,i}^m$
$IF_{t,j}^M$	imbalance ratio parameter of $\Psi_{t,j}^M$
K	maximimum number of clusters
k	number of clusters
K_r	kernel function
λ	Poisson distribution parameter
$\mathcal{LS}$	learning set
$\mathcal{LS}^m$	minority data learning set
$\mathcal{LS}^M$	majority data learning set
$\mathcal{M}$	set of class labels
n_s	number of samples
n^m	number of minority class samples
n^M	number of majority class samples
N_s	number of samples stored in ADS
n	number of minority clusters
N	number of majority clusters
ν	parameter regulating the number of outliers and support vectors

p	number of neurons in layer
P_b	bootstrap probability function
P_t	probability of selecting samples from the t-th data chunk
Π	a pool of base classifiers
Π^m	a pool of minority classifiers
Π^M	a pool of majority classifiers
Ψ	classification algorithm
$\Psi^m_{t,i}$	i-th minority model trained on the t-th data chunk
$\Psi^M_{t,j}$	j-th majority model trained on the t-th data chunk
ϕ	mapping function
R	fraction randomly selected samples
ρ	margin of the SVM hyperplane
s	maximum size of classifier ensemble
S	maximum number of stored data chunks
$S^m_{t,i}$	predictive quality parameter of $\Psi^m_{t,i}$
$S^M_{t,j}$	predictive quality parameter of $\Psi^M_{t,j}$
$Score$	drift detector score
$Score_L$	drift detector scores list
$Score_M$	drift detector mean score
SC	silhouette coefficient function
SV	silhouette value function
T	threshold parameter
t	current timestamp
t_i	number of algorithm iterations
w	normal vector to the SVM hyperplane
x	feature vector

$x^{(l)}$	l-th feature
x_i^k	k-th example in the i-th data chunk
$\mathcal{X}$	feature space
X_i	i-th constituent of $\mathcal{X}$
$\mathbf{X}$	random variable
ξ	slack variable
y	class label vector
$w_{t,i}^m$	weight of $\Psi_{t,i}^m$
$w_{t,j}^M$	weight of $\Psi_{t,j}^M$

Contents

Chapter 1

Introduction

Stanisław Lem is a writer of science fiction novels that inspire me. The fiction described in his books is slowly becoming the reality that surrounds us. In 1951, in one of his first novels "The Astronauts" [161], he wrote about a network connecting machines that worked together and thanks to that they were much more efficient. This idea can be compared to the Internet, which was created at the turn of the 1960s. In 1955, in the novel "The Magellanic Cloud" [162], Lem equipped the crew of the spaceship "Gea" with a "crystal record" medium, on which any information can be written. Today, it can be perceived as floppy disks, CDs, or flash drives, which were created many years later. In 1961, Lem wrote in the novel "Return from the stars" about a kind of book that today can be called an e-book reader. *"The books were crystals with recorded contents. They could be read with the aid of an opton, which was similar to a book but had only one page between the covers. At a touch, successive pages of the text appeared on it."* [164]. In 1964, in the story "Tale of the Computer That Fought a Dragon" of the novel "Mortal Engines" [163], the King Poleander Partobon, the ruler of Cyberia, asks the "machine" what to do to defeat the dragon. The vision reminds me quite strongly of current systems or machines that help humans make decisions. Not necessarily a way to get rid of the dragon, but something more mundane.

The world of science has also had visions that machines might one day equal humans abilities. In 1950, Alan Turing began his work with the sentence, *"I PROPOSE to consider the question, 'Can machines think?'"* [251]. His work describes the proposed Turing test, also called the "Imitation Game". The goal of this test was to experimentally obtain an answer to the question posed above. Difficulties in clearly defining the meaning of the word "thinking" resulted in an unwritten change of this question to *"Can machines do what we as thinking entities can do?"* [107].

Time has shown that the answer to the second question is partially affirmative. Present technologies based on artificial intelligence systems are applied in many areas of human life. It helps browsing various web resources with intelligent search engines [125]. It makes easier financial market analysis [45], which helps in making some decisions in the financial sector. It can help us to choose or suggest an interesting offer using the product recommendation system [76]. The route to work can be determined and planned to avoid the estimated traffic jams on the way [169]. Also our email inboxes have sophisticated systems to filter incoming messages [15]. One can say that some possibilities provided to us by these technologies become necessary for normal functioning in the modern world. Sometimes machines or applications surpass human capabilities in certain activities.

There are many science fields that support the hypothesis of machines' superiority over humans in selected life areas. One of them is machine learning [188]. It is a certain subfield of artificial intelligence. The dissertation will focus on one branch of machine learning - classification [270]. In simple words, the classification task focuses on solving the problem of matching objects with labels [7]. Objects have a description in the form of attributes and assigned labels to denote class membership. Classifiers, on the basis of previous observations, try to model the dependence of objects' membership [27]. There are many different classification systems dedicated to special related problems. One such problem is the classification of imbalanced data.

Most of the data usually does not have an even class distribution. However, the problem arises when these differences are significant [79]. This phenomenon is called data imbalance [140]. Most often, this is associated with binary problems, but it also occurs in multiclass problems [263]. For binary classification problems, the majority class, the dominant one, is called negative and the minority class is called positive. In practice, the highly unequal distribution causes standard classifiers to overclassify objects as majority [248]. This results in a poor classification ability of positive samples. In addition, a major problem that accompanies this type of data is conducting a robust evaluation [40]. Data imbalance affects common metrics in a highly negative way. Even the aggregate measures that are generally used do not seem to be the ideal solution [106].

It is worth noting that in imbalanced data, most often the minority class has a very high misclassification cost [112]. A well illustrative example would be the COVID-19 virus diagnosis system. Healthy people, which are the majority set, do not create any threat and will not spread the virus. The sick persons are in a significant minority. They are much more important to identify. The standard classifier used in such a system will get more examples of healthy people, thus it may bias towards the majority class - a diagnosis as a patient without the virus. Applications that face these difficulties are fault detection [289], anomaly detection [176], medical diagnosis [265], spam filtering [2]

to enumerate only a few. Imbalance can be nullified using preprocessing methods to balance the data or using algorithms that have special heuristics to improve minority class classification [218].

In recent times, there has been a continuous growth in the amount of data produced by people. One of the reasons behind this is the extensive usage of information technology in many circumstances. This is fostered by social media, VOD services, widespread voice and text communication, mobile devices, smartwatches, smartbands, which often record our activities and basic life functions. All of these contribute to the production and consumption of various types of data in large amounts. Research in the project "DATA NEVER SLEEPS 6.0" [1] by the DOMO company estimates that in 2020 *"for every person on earth, 1.7 MB of data will be created every second"*. The world is recently confronting the COVID-19 pandemic, which is forcing even more people to work, study, or handle daily activities remotely. This additionally increases the estimated values.

To cope with this situation, the term *big data* was introduced to describe large, variable, and diverse datasets that are inherently difficult to process and analyze, but the knowledge that can be extracted from them is valuable [274]. The 3V model was proposed defining *Big data* as [199] [158]:

- Volume – large amount of data.
- Velocity – high speed of data processing.
- Variety – diversity of data.

Later, more characteristics have been added as Veracity or Value. One of the examples for this type of data are streams. In machine learning, data streams are rapidly incoming high volume pieces of information. Theoretically, the data stream can be infinite [85]. One of the main problems that arises in the classification of this type of data is the occurrence of concept drift [88]. It also known as non-stationary data stream. This means that the incoming data starts to make some fluctuations that significantly affect the predictive performance of the trained classification models [26]. The non-stationary data stream problem can be well described using the recommendation system as an example. This kind of system provides products or services to users that are possibly interesting to them. Prediction is mainly based on the history of buying or browsing the web. Users' interest in products and services changes for reasons such as current needs, prevailing trends, user age, etc. Building a model once and expecting it to be reliable in the future is wrong. Therefore, the model of the recommendation system, which provides predicted offers, must be able to adapt to the changing user's needs.

[1] https://www.domo.com/assets/downloads/18_domo_data-never-sleeps-6+verticals.pdf

Non-stationary data stream classification requires special treatment. It is important to identify the concept drift and counteract this phenomenon [175]. Drifts cause some disturbances resulting in outdated models. To prevent this, different strategies can be employed - drift detection [13] or incremental learning [74]. There are many additional difficulties in data stream classification. Some work deals with the problem of label availability [146]. Other work focuses on the problem of limitations in memory and computational resources [147]. It means that special attention should be paid to ensure the ability of algorithms to process the data relatively quickly along with maintaining certain memory constraints [208]. At the same time, the data stream is a very valuable source of information that should not be excluded due to its demanding nature [85].

One well-known solution to improve the classification quality of difficult data is forming classifier ensembles [286]. The ensemble is usually composed of several "weak learners". However, their combined decision greatly gains on the quality [252]. This happens due to the good generalization ability of the whole committee, which has an advantage over one "strong learner" [294]. For proper performance, it is necessary to ensure that the models are well diversified and the sophisticated combination rule is designed [67]. Classifier committees perform very effectively for solving difficult problems such as non-stationary drifting streams [98, 143] and strongly imbalanced data [86]. Looking from the practical side, approaches based on ensembles of classifiers can be found in many applications such as: spam detection [173], face recognition [177], cancer diagnosis [151], fault diagnosis [124], text categorization [203].

Imbalanced data combined with streams significantly increases the difficulty of the problem [35]. Most often the classification of data streams and imbalanced data is considered separately. The conducted literature study shows that there are few works that deal with the problem of classifying data streams that have an imbalanced class distribution [160]. For non-stationary imbalanced data streams, drift can also imply variation in the imbalance ratio, which can be called dynamic imbalanced data [79]. A very special case of such a change is the temporary or complete disappearance of objects from one of the classes. For such a circumstance, the classification method based on one-class classifiers seems to be a good proposition [75]. It is hard to find solutions that use one-class classifiers for this type of problem. When data is imbalanced, a widespread solution is to use data preprocessing [93]. It is worth noting that most approaches for preprocessing imbalanced data streams do not rely on data accumulation from previous chunks, but use prototype selection or artificial sample generation methods [93]. Therefore, in the dissertation I will focus on proposing new approaches to imbalanced data stream classification that employ one-class classifiers and data preprocessing methods based on data accumulation, because there are no, or very few similar solutions. In relation to the above, a research hypothesis was formulated:

One may design ensemble methods that use data sampling techniques and one-class classifiers, which can outperform state-of-the-art ensemble algorithms for imbalanced data streams classification.

To validate the research hypothesis stated above, the following research objectives have been formulated:

- *Designing a one-class classifiers ensemble method to solve the problem of imbalanced data stream classification.*
- *Extending the one-class classifiers ensemble method by introducing an improved weighted decision rule for better adaptation to imbalanced data streams.*
- *Proposing a classifier for imbalanced data streams with hybrid data accumulation sampling technique.*
- *Upgrading the classifier with data accumulation a hybrid sampling to a weighted bagging ensemble for imbalanced data stream classification.*
- *Improving the imbalanced data stream classifier with hybrid data accumulation sampling to ensemble method with concept drift detector.*
- *Designing an ensemble method for imbalanced data stream classification that combines hybrid data accumulation sampling and one-class classifiers.*

The structure of the dissertation will be outlined below. Chapter 2 will present topics related to the dissertation. Basic concepts associated with machine learning will be discussed and imbalanced as well as streaming data classification task will be described in more details. In chapter 3 methods for classifying imbalanced data streams based on building ensembles of one-class classifiers will be proposed. Two different approaches will be presented and evaluated along with a comparative analysis. Chapter 4 focuses on methods that use hybrid data preprocessing techniques to perform imbalanced data stream classification. Four different methods will be proposed that attempt to perform semi-synthetic oversampling based on collected samples from previous data chunks. Chapter 5 summarizes this work by answering the previously stated research objectives.

Chapter 2

Selected topics of pattern classification

This chapter focuses on introducing the topic of this dissertation. A variety of issues related to multiple domains of machine learning are discussed as well as basic concepts related to the classification task. Then, topics such as ensemble learning, one-class classification, imbalanced data, and data stream learning are presented, along with a description of selected methods that address these problems.

2.1 Machine learning

Machine learning is an artificial intelligence branch that focuses on creating systems that use prior knowledge to improve, develop, and achieve certain goals without being directly programmed to accomplish them. There are several different definitions for machine learning. In 1959, Arthur Samuel defined machine learning as a circumstance where "A computer can be programmed so that it will learn to play a better game of checkers than can be played by the person who wrote the program." [222]. Tom Mitchell proposed it as “A computer program is said to learn from experience (E) with respect to some class of tasks (T) and performance measure (P), if its performance at tasks in T, as measured by P, improves with experience E” [188]. Ethem Alpaydin defined it as "Programming computers to optimize a performance criterion using example data or experience" [7]. "Machine learning is an area of artificial intelligence concerned with the study of computer algorithms that improve automatically through experience. In practice, this involves creating programs that optimize a performance criterion through the analysis of data" was proposed by Martin Sewell [229]. In other words, machine

learning algorithms can produce a program capable of solving a problem without using a directly programmed solution, but based on the knowledge gained.

It may be assumed that machine learning algorithms try to mimic in some degree the behavior of human beings who, through learning, acquire knowledge and information as input data, try to achieve a certain goal [188]. One such goal could be pattern classification, which our brain solves many times in a day. By looking at the objects around us, we can very easily identify what we see, name it, and place it into some predetermined category. An example would be an apple and a pear. Each apple and pear is unique in its way. It has a different shade or color of skin, size, shape, freshness, and aroma, but still, without precise instructions, we can distinguish between these two fruits. Similarly, the machine, without the exact code of the program that solves the problem [242], will be able to distinguish between the two fruits based on previous examples of apples and pears. Most importantly, the good model should be able to recognize an apple or a pear that it has never seen before.

When face a new problem, one of the good ideas is to work on problem identification and definition. One of the optimal ways to solve such tasks is to build models which, collecting essential information from the data, can work out their own methodology to solve a certain task [159]. Machine learning deals with various approaches to solve these types of problems. Four different machine learning schemes can be distinguished: supervised, unsupervised, semi-supervised, or reinforcement learning. Each of these subfields is highly dependent on the characteristics of the data describing the issue under consideration and the expected result that the program should obtain [7]. At the same time, each of these approaches has completely different assumptions. Machine learning involves the technological memorization of certain patterns and behaviors that a machine has already "experienced" in the past [121]. In addition, it also makes it possible to analyze new data and look for solutions that have worked in other situations previously known to the machine.

Supervised learning is one of the machine learning subfields, which deals with issues where having input variables (training data) and a certain output variable, it is possible to create a mapping function from input to output [27]. Instead of manually setting precise rules for performing prediction, supervised machine learning relies on using a certain set of examples [7]. These examples should have correct outcome values for each of them. These data could be used to "train" a method to automatically predict the answer in both the examples used for learning and other examples not previously known. The user who enters the output data is a kind of supervisor who directs the learning algorithm to correct answers so that eventually the algorithm can return them on its

own [270]. Two subcategories of supervised learning can be distinguished - classification and regression. The former focuses on matching objects to classes based on given data [3]. Having learning data, the algorithm is able to build a certain model describing the dependencies of objects from particular classes and is able to distinguish them. In addition, to teach the prediction of correct labels in a classification problem, supervised learning can also be used in situations where the predicted outcome is a number. The latter assumes that the output data describes not the membership of objects to classes, but certain values [77]. Formally, regression is a statistical method that allows estimating the conditional expected value of a random variable, called the explained variable. This is done by building a model or function that describes how the expected value of the explained variable depends on the explanatory variables - the input data.

Unsupervised learning uses data that does not have labels describing the class membership of objects [109]. The system examines the data and tries to find some regularity inside. The main task that the algorithm wants to perform is to identify patterns and correlations among the data samples [101]. Looking at this problem from a broader perspective, unsupervised learning resembles the way humans perceive the world around them. We use intuition and experience to connect some events into groups. As we gain more experience, our ability to better categorize or identify certain concepts grows. Experience gained by machines is defined by the amount of input data. This is especially useful for analyzing very large databases, among which it is difficult for a human to find dependencies on their own.

Semi-supervised learning combines the previous two schemes of learning. Usually, data that is produced during various processes do not have proper labels, or these labels are incomplete [293]. The labeling process itself can be quite costly and human resource intensive [47]. For this reason, one solution is to use a semi-supervised learning approach that solves tasks with both labeled data inputs (containing corresponding outputs, specific examples) and unlabeled data inputs (requiring mapping to the outputs, finding answers) [297]. Based on the data described by the labels and using residual analysis of the unlabeled data by unsupervised learning methods or other techniques, the algorithm attempts to find appropriate relationships among the samples. There are also self-labeling techniques that allow to automatically fill in missing labels in processed datasets [99]. Active learning is a very special case of semi-supervised learning in machine learning [4]. This is very useful when the cost of labeling is high, or for time reasons it is not possible to label all instances. The method for active learning must indicate for which samples to take labels so that it selects only those data that have high value for solving the problem. The main idea is that if a machine learning algorithm can select the data on which it wants to build a predictive model, it can perform better than traditional methods with much less data used for training [228].

Reinforcement learning is a task which was initially formulated by Richard Sutton [212]. It is when a system operates in an unknown environment. There is a lack of both specific input and output data. The only information the learning machine receives is a so-called reinforcement signal. This signal can be either positive, which can be described as a reward, or negative, which can be described as a punishment [127]. The machine receives some ready-made set of allowed actions, rules, and statements. Acting within their framework, it analyzes and observes their effects. It uses the rules in such a way as to achieve the desired effect. This method can be otherwise called the trial-and-error method, as the system makes certain decisions that are evaluated later [266]. An example would be playing a new game where we do not know the exact rules, but we know how to make our moves. After the game is over, we determine whether we won or lost. We come out better in subsequent games because we start to build our own strategy based on the results of previous tries.

2.2 Classification

Classification is one of the tasks in supervised learning, where the output variable is a label with the class name of a given object [7]. The topic of classification will be discussed much more extensively, as it is the main problem that will be considered in the dissertation. Classifiers are algorithms that, based on training data, create models capable of indicating the class membership of as yet unrecognized objects. They learn the structure of a data set containing examples separated into groups with labels. Then, the model receives unseen data and tries to classify new examples into appropriate classes [3]. Sometimes this process can be made in such a way that several sets of classification rules are generated, and then the optimization process selects the ones that best meet the posed criterion [18]. Denis Michie defines classification as "a learning system that uses sample data (the training set) to generate an updated basis for improved classification of subsequent data from the same source" [182]. Although this definition is mainly concerned with the classification problem it is also considered by the author as a good definition of machine learning.

The mathematical formulation of the classification [270] could be presented as follows. Let us denote $\mathcal{X}$ as a feature space and x as a feature vector, so an example is described by attribute values. Feature set having d attributes:

$$x = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \cdots \\ x^{(d)} \end{bmatrix}, \text{ and } x \in \mathcal{X} = \mathcal{X}^{(1)} \times \mathcal{X}^{(2)} \times \cdots \times \mathcal{X}^{(d)} \tag{2.1}$$

An essential fact is that most of the data involved in solving a classification task may have attributes that consist of continuous numerical values. However, it is worth noting that it should not be explicitly assumed that all attributes have numeric values. One should still distinguish between nominal and categorical attributes, which are those that consist of a set of repeated values. Such data should be processed using selected approaches that transform from categorical to binary data or numerical data. Therefore, it can be assumed that the feature vector belongs to the set of real numbers:

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

As mentioned earlier, the main task of classification is to find the membership of objects to given classes specified by the feature vector. These classes are described by a certain set of labels, which can be denoted as $\mathcal{M} = \{1, \ldots, M\}$. In a formal way classification is presented as a function Ψ where $\mathcal{X}$ is its domain and $\mathcal{M}$ – codomain.

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

The main goal that the learning algorithm tries to achieve when training a new model is to minimize the cost of an incorrect decision. This leads to the introduction of a loss function to be minimized, but this requires from the user the definition of an associated loss function. Since the user usually cannot formulate this correctly other alternate criteria are used. This most often leads to the use of zero-one evaluations and error cost minimization based on classification quality metrics. Typically, the decision made by the classifier is primarily based on the supports that can be returned by the model. Canonical model of the pattern classification assumes that there is a set of discriminative functions. These are a type of functions that determine the support of a given class, which can be expressed using distance measure or probability, depending on which classifier is used. For example, it could be the output of an artificial neural network. Making a final classification decision requires the use of a function indicating maximum from the resulting model response, or a support set:

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

Classification issues

The classification aims to make the predictive performance of the model as good as possible. A classifier learns patterns and predicts unknown sample labels with the highest predictive performance [27]. This process focuses, among other things, on optimizing a chosen performance metric. Usually as the accuracy or classification error are used as the learning criterion. It should be noted that three error components could be distinguished. The first one called bias error describes how much the algorithm overgeneralizes its predictions. This means that a model that generalizes too much may be able to miss the relevant relations between features and target outputs [65]. On the other hand, important is model variance error, which determines how well the model is fitted to the data that was used for learning. The third error that should also be mentioned is the variance of the irreducible error in the data. This error is mainly related to the irregularities that occur in the data such as missing values, labels noise, feature noise, etc. It is an error that cannot be corrected by optimizing the model. These three errors contribute to one of the most important problems that lie behind supervised learning as well as classification, called bias-variance dilemma [64].

Another major problem is overfitting. It is a phenomenon that occurs when a model is too complex [223]. This problem may be magnified both when there is a large variance in the training data and when the model is too flexible. Models should be built according to the principle of Ockham's razor – "Plurality must never be posited without necessity". When the model's degrees of freedom exceeds the data's information content, parameter selection becomes mostly a coincidence. In the classification task, overfitted models can predict the learning data very well, but they will perform worse when applied to data that they have not encountered during learning. This means that the model starts to fit random errors in the training data gaining greater accuracy for that data, and thus its generalization ability fades away [171]. Accuracy is the concept of measuring how well a model has learned to solve a specific task. It will determine the ability to solve already known tasks, but generalization will indicate the ability to solve new tasks similar to those it knows. It is not possible to identify one best solution, it is always some trade-off between the data variance and bias [31].

There are theoretical studies dealing with this issue [259], but from the point of view of practical applications, it is dropped to the stage of experimental evaluation. An example way to eliminate overfitting is to use a cross-validation procedure. It involves dividing

the data sample into subsets, performing an analysis on the learning set, and validating the analysis on the test set. The purpose of cross-validation is to test the model's ability to predict new data that was not used to estimate it, to determine whether overfitting will be a problem for the model [282]. In situations where cross-validation indicates very poor results despite the model learning process being performed correctly, it may also be a case of the model underfitting. This occurs when the learned model is too simple (not enough observations or features) and therefore does not learn well from the training data.

Another issue is the classification of objects in high-dimensional spaces, in which each object is described by tens, hundreds, or even thousands of attributes. This phenomenon is called the curse of dimensionality [257]. The term was probably used first by Richard E. Bellman [20] in the context of optimization over a large number of variables. Such multidimensional sets are encountered increasingly frequently in real-world datasets. According to the Hughes phenomenon [118], when the number of dimensions is increased while keeping the sample size unchanged, the model performance decreases significantly. It results from the fact that in a multidimensional space, the similarity of objects in this space comes closer to each other [68]. It means that to properly divide the set into classes, one needs much more objects. Meeting this condition is often difficult or even impossible, because there is no unlimited set of data. To ensure the same classification accuracy as in the case of space with a smaller dimension, the number of samples, i.e., the size of the data, must grow exponentially as the dimension of the feature space increases [27]. In addition to the need for increasing data, the number of possible feature variants grows exponentially, which significantly increases the computational complexity of algorithms.

Due to the curse of dimensionality, correct classification of objects may prove to be completely impossible if one decided to use the full set of attributes. For this reason, efforts are made to reduce the dimensionality of the feature space by using various methods of feature selection and information reduction in the classification task [129]. It is worth noting that in most cases, objects are generally well-defined using only a certain subset of attributes. However, it must be taken into account that while reducing the dimensions, i.e., the number of features taken into consideration during classification, some information may be lost [34]. Nevertheless, the problem becomes correct identification of such a subset of attributes, which will enable the classification algorithm to correctly recognize objects and assign them to appropriate classes.

Such a subset of attributes may be created by selecting a certain number of attributes from all the possessed features, or by extraction, that is in short, creating new attributes, but based on a certain set of existing attributes.

– **Feature selection**

The goal of feature selection is to choose from all available features such a subset of attributes, which contains only those ones that are significant from the point of view of the problem under consideration [42]. In other words, selection can be understood as removing redundant features that are irrelevant to the specific task. The mentioned subset should contain the minimum number of features that have the greatest impact on the quality of the model, that is, on the highest possible classification accuracy. In this way, the discussed irrelevant and redundant features are discarded.

$$x = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \cdots \\ x^{(d)} \end{bmatrix} \rightarrow \bar{x} = \begin{bmatrix} \bar{x}^{(1)} \\ \bar{x}^{(2)} \\ \cdots \\ \bar{x}^{(d')} \end{bmatrix}, d' < d \tag{2.5}$$

– **Feature extraction**

It consists in determining completely new attributes, however, those that are functions of the original attributes, i.e., they depend on the features that the objects have [129]. This allows to significantly reduce the dimensionality of the problem and to cope with the curse of dimensionality. Very often it also turns out that a smaller number of new attributes contains almost the same amount of information about objects as a much larger number of original attributes. Additionally, the classification algorithm is far less prone to overfitting.

$$x = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \cdots \\ x^{(d)} \end{bmatrix} \rightarrow \bar{x} = \begin{bmatrix} x'^{(1)} \\ x'^{(2)} \\ \cdots \\ x'^{(d')} \end{bmatrix} = f\left(\begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \cdots \\ x^{(d)} \end{bmatrix} \right), d' < d \tag{2.6}$$

Selected classification algorithms

Several types of classification models can be specified, which build their predictive ability in different ways depending on their main assumptions. There are many classifiers, but the description will be limited to a few that will be used further in the dissertation during:

– **Support vector machine**

It is one of the most popular kernel machine methods proposed by Vapnik [256]. It is often used because of several advantages [7], such as Vapnik's principle, that the actual problem should be solved before the complex problem or the asset of support vectors which lie close to the boundary and some information can be obtained from them in classification such as generalization error or the model parameter. Fig. 2.1 presents the basic SVM model in binary classification. The main task is to find optimal separating hyperplane with the maximum margin which divides examples into two sets [272].

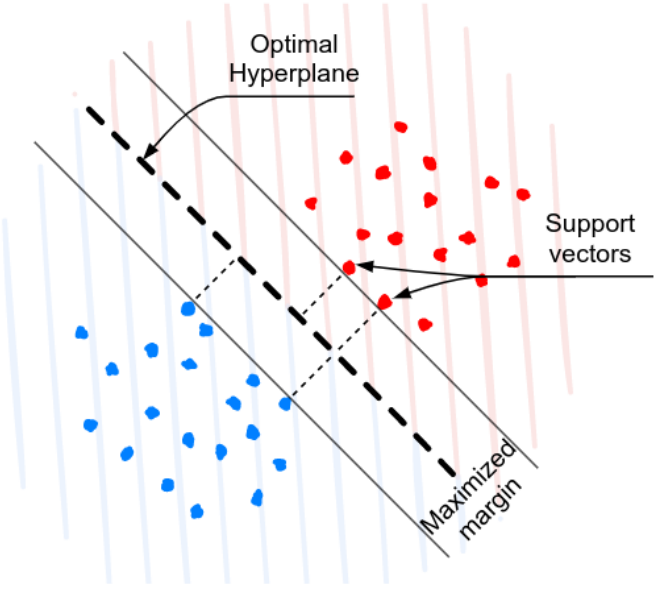


Figure 2.1: *SVM model in the binary classification*

– **Naïve Bayes classifier**

A Naïve Bayes classifier can be used to determine the probability of classes based on a set of patterns [213]. It bases its main assumption on the fact that the variables describing the features are conditionally independent of each other, given a class. This idea is built on the Bayesian principle. Despite its naive nature, this method has some classification ability. The most common model using this idea is the Gaussian Naïve Bayes [120]. The model introduces some a priori assumptions that the data distribution is close to a Gaussian distribution. Fig. 2.2 shows a simplified diagram of the classification process using the Naive Gaussian Bayes algorithm. The black point represents the sample to be classified. Red represents class 1 and blue represents class 2. The hatched areas indicate the probability distribution of the class, and the dashed line ending with a cross at the bottom indicates the average value. The distance between the sample and this point is the

z-score, which is the value that determines the probability of the sample belonging to the given class.

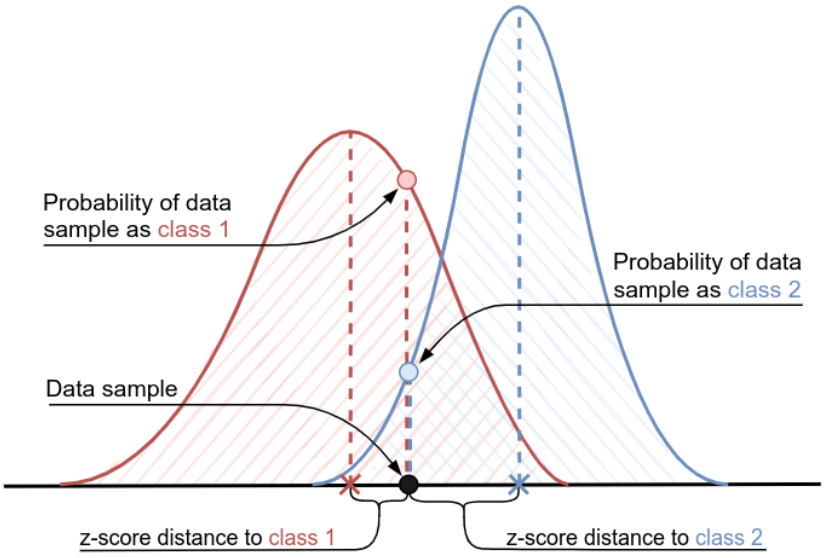


Figure 2.2: *Guassian Naïve Bayes classification diagram*

– **Decision tree**

A decision tree is a nonparametric, hierarchical model beginning at the root and finishing at the leaf. Each decision node contains test function which is applied on the input data, and the output as branches of discrete values. The lower level has the next decision node or the leaf terminating the algorithm by giving decision or the class label. The decision tree breaks down complex problem into a set of simple rules IF-THEN which is easy to interpret [27]. An example would be a tree deciding whether to go play golf. In Fig. 2.3, you can see that the data has 3 features that influence the decision – Outlook, Windy, and Humidity. Depending on what weather conditions are currently met, the model will lead the user through one of the paths. At the end of each is a decision which is a class label. The most popular trees used in the classification are C4.5 [207] and Classification And Regression Tree (CART) [30].

– **Multi-layer perceptron**

This model includes many individual perceptrons in each the input is multiplied by weights and the output is the classified data. The first perceptron proposed by Rosenblatt [215] checks the threshold and assign the output as 1 or -1 using linear function. The function can be also nonlinear, e.g. Heaviside step function, Continuous Log-Sigmoid function, or Continuous Tan-Sigmoid function [270]. *MLP* has at least 3 layers (Fig. 2.4) where the output of the first perceptrons' layer is the input of the next one, and the perceptrons are not connected within one layer.

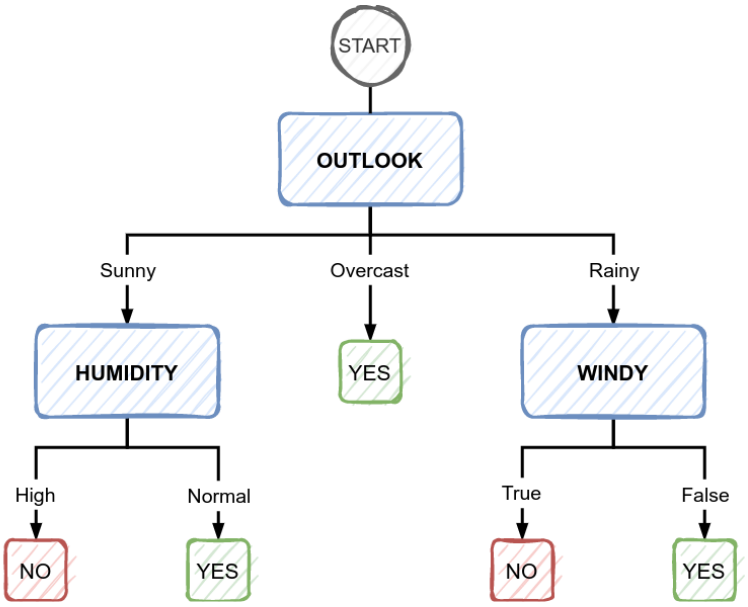


Figure 2.3: *Decision tree model example on weather data*

Input, hidden and output layers contain information about weights and try to minimize the error [110].

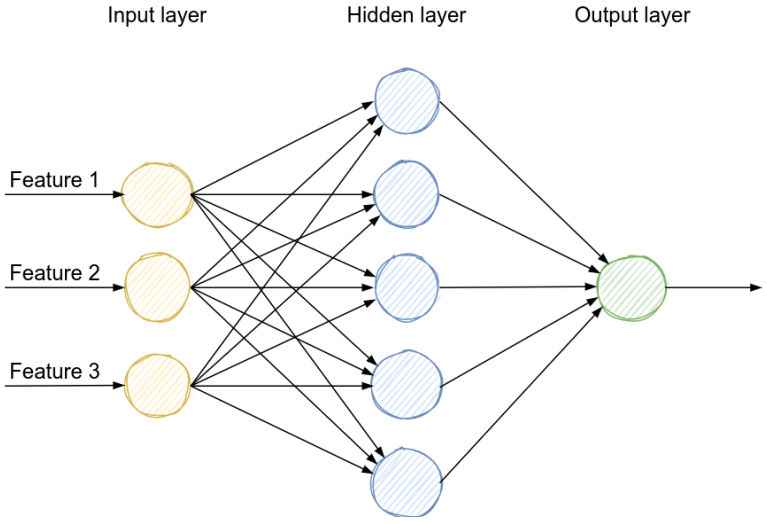


Figure 2.4: *Multi layer perceptron structure diagram*

– **Minimum distance classifiers**

These are models that mainly use pattern distance during classification. The best known is the k nearest neighbors (KNN) classification algorithm [55]. It is characterized by two particular features - non-parametric and "lazy". The former means that this algorithm does not assume in advance that it is dealing with a certain distribution of data. This is a very useful assumption, because in the real world, data are often not linearly separable or do not map very well to a normal or any

other distribution. "Lazy", on the other hand, means that the algorithm does not build a generalizable model of the problem in the learning phase. One can say that learning is deferred until a pattern is fed into the model that needs to be classified. In the Fig. 2.5 is presented an example of classification using 1-NN or 3-NN version of this model. Depending on the k neighbor number, the final decision may differ.

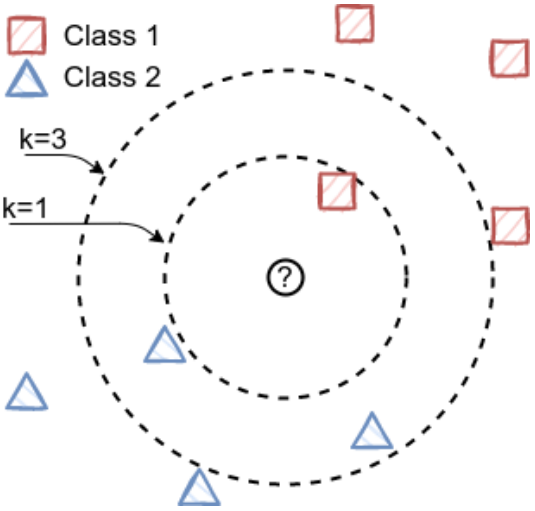


Figure 2.5: *K nearest neighbor model example*

2.3 One-class classifiers

The main idea of the one-class classification allows learning a new model using data without counterexamples or when obtaining them involves a high cost. It means that the classifier can generate a model based on objects from only one class. An example of such an application is the system to analyze the efficiency of the ship's engine [292]. The designed system should have the ability to show a certain probability of malfunction. The moment of failure and the acquisition of data that will determine such a state may be associated with substantial losses that will have to be suffered. Applying a one-class classifier to this type of problem enables designing such a system without capturing failure events. One-class classifiers are also suitable for anomaly detection [75]. Such a problem focuses on detecting very rare observations or events that are not specified at the beginning of the learning process but differ significantly from most known observations.

Among the one-class classification methods, One-Class Support Vector Machine ($OCSVM$) has become the most popular approach. One of the first ideas is Support Vector Domain Description [245]. It is an algorithm that creates a hypersphere around the objects from one class and tries to close all instances in the smallest area. Then Scholkopf et al. [226] proposed to push the decision limit a little further away from the samples. This method

allows for different kernel transformations. Fa Zhu et al. [296] proposed a new strategy for weighted one-class support vector machine. The idea is to increase the weight of the samples closer to the center of the training dataset and reduce the weight of the more distant ones. Xing et al. [277] presented the robust sparse coding based on cross entropy and logarithmic penalty function (RSCCLPF) for *OCSVM* classification problems.

The general idea of OCSVM is similar to the SVM classifier, but it is transformed to the one-class hypersphere minimization. In the basic concept, the SVM classifier algorithm tries to solve the following problem:

$$\begin{aligned} &\underset{w,b,\zeta}{\text{minimize}} \ \frac{\|w\|^2}{2} + C\sum_{i=1}^{n_s}\zeta_i \\ &\text{subject to } y_i(w^T\phi(x_i)+b) \geq 1-\zeta_i, \\ &\zeta_i \geq 0, i = 1, ..., n_s \end{aligned} \tag{2.7}$$

where $x_i \in \mathbb{R}^p$, is training object in binary problem, $y \in \{1,-1\}_s^n$ is labels vector and $(w^T\phi(x_i)+b)$ is the separating hyperplane. ζ_i is distance of samples outside the correct boundary and C defines the strength of this penalty. When this minimization problem is solved with Lagrange multipliers, the decision function for a data sample x becomes:

$$f(x) = \text{sgn}(\sum_{i=1}^{n_s}\alpha_i y_i K_r(x, x_i) + b) \tag{2.8}$$

where $K(x, x_i)$ is a kernel function and α_i is the dual coefficients. The OCSVM method proposed by Scholkopf et al. [225] try to solve:

$$\begin{aligned} &\underset{w,\ \xi_i,\ \rho}{\text{minimize}} \ \frac{\|w\|^2}{2} + \frac{1}{\nu n_s}\sum_{i=1}^{n_s}\xi_i - \rho \\ &\text{subject to } (w\cdot\phi(x_i)) \geq \rho - \xi_i \text{ for all } i = 1, \ldots, n_s \\ &\xi_i \geq 0 \text{ for all } i = 1, \ldots, n_s \end{aligned} \tag{2.9}$$

where ν defines an upper bound on the fraction of outliers and lower bound on the number of training examples used as Support Vector. After solving above equation using Lagrange multipliers the OCSVM decision function is equal to:

$$f(x) = \sum_{i=1}^{n_s} \alpha_i K_r(x, x_i) - \rho \tag{2.10}$$

Such a function defines the distances of samples to the model decision regions. Its negative value is for objects outside the classification borders, and positive value for an object inside:

$$\psi(x) = \text{sgn}(f(x)) \tag{2.11}$$

However, it is worth noting that there are also proposals based on other classifiers. Khan et al. [130] described and compared the implementations of different varieties of one-class nearest neighbors classifiers. The authors distinguished four types of this method. The *11NN*, which looks for the first pattern belonging to the class, and then calculates the distance to the first nearest neighbors of this pattern. Variant *J1NN*, which works similarly to the previous one, except that several patterns are searched and the average value is calculated. Approach *1KNN*, where only one pattern is searched, but the distance to several neighbors is measured. The next is *JKNNN*, which combines rule from both previous and searches for multiple patterns and multiple neighbors. A certain threshold determines whether the classified sample belongs to a given class or is an outlier based on the calculated distances. Another idea is to use decision trees for one-class classification. The one-class decision tree [119] tries to find divisions in all features where objects from one class are located. There are also some methods based on one-class random forests [60]. Ruff et al. [220] have proposed an idea that uses artificial neural networks with some assumptions of the SVDD [247]. The data is transformed to be in the smallest possible hyperspherical area.

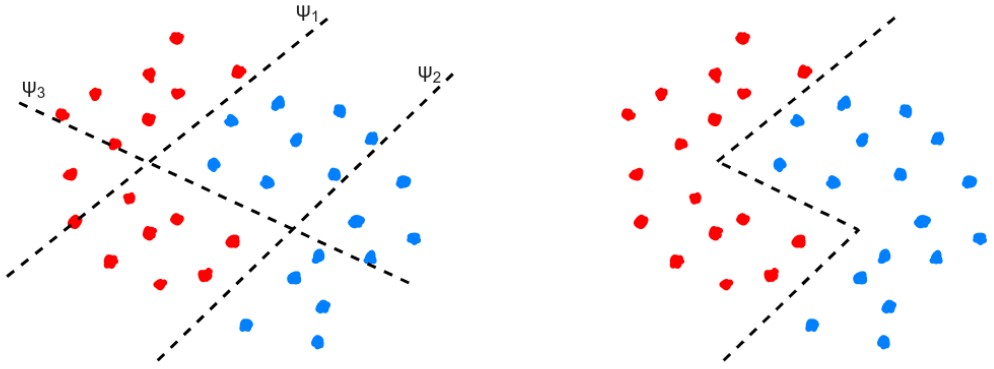
2.4 Ensemble learning

It is difficult to determine the exact genesis of the ensemble learning or ensemble decision. The origins of this idea date to very distant times, where methods of decision making at the state level were used. Democracy, which in the literal translation is the people' rule, can be used as one of the examples. It is a system established in ancient Greece, and more specifically in Athens during the reign of Solon, which provides that the supreme power in the state to be carried out by all citizens who have political rights. Another real-world example of the ensemble decision making can be found in court. A jury is

an institution of justice found most commonly in countries with a *common law* system. Jury is composed of ordinary citizens, not professional judges with expertise in law. Adjudication is based on answers (affirmative or negative) to questions posed by the professional judge presiding over the trial. On the correctness of these ideas, Marie Jean Antoine Nicolas de Caritat, Marquis of Condorcet in 1785 formulated *Condorcet's jury* theorem in his work *Essay on the Application of Analysis to the Probability of Majority Decisions*. In general, this work is concerned with the problem about the probability of voters independently making mistakes and the relationship of increasing and decreasing the voters' number [22]. The jury contains many analogies that relate to ensemble learning ideas.

Returning to the dissertation field of study, it is necessary to clearly define what ensemble learning is in the context of classification and machine learning. In 1984, Valiant [253] proposed Probably Approximately Correct (PAC) learning. It is a framework in which the learner obtains random samples based on which it has to choose a generalization function from among various available ones. Then in 1989, Kearns and Valiant [128] proposed the concept of weak learnability. In their work, they ask the question, "Can a set of weak learners create a single strong learner?". In 1990 Shapire [224] affirmative answered and developed these concepts. Strong learning means the classifier has arbitrarily good accuracy. In contrast, weak learning means the classifier performs slightly better than the random guessing model. Forming a weak learner is a much simpler task. These ideas are the foundations for ensemble learning.

The most important reason that positively supports the use of ensembles to solve classification problems is the ability to increase the predictive performance of weak models [32]. The main assumption of ensemble learning is that one main classifier, called "ensemble", is composed of many weak learners. However, the joint decision made by them can be better or equal as one strong learner [286]. For example, there is some data that is not linearly separated in any way - two interleaved half circles. No linear classifier will be able to prove error-free prediction (Fig. 2.6a). However, using an ensemble of classifiers that consists of linear models can solve this classification problem very easily (Fig. 2.6b). According to Wolpert's theorem [268], there is no single method that will always work best under different conditions. Another reason which supports the idea of classifier ensembles is that this approach sometimes gains a little more resistance to model overfitting than a single model [205]. A well-constructed ensemble is a model more robust to this type of phenomenon due to the fact that the final decision is an average of the models' outputs [148]. Additionally, another noteworthy aspect is the considerable parallelization capability of the ensemble learning classification algorithm [254]. Confirmation of the ensemble learning idea can also be sought in many fields including the realms of social life [221].



(a) Classifiers *(b) Ensemble*

Figure 2.6: *Decision regions visualisation of linear classifiers and ensemble of linear classifiers*

Methods based on the idea of ensemble learning place certain requirements on those who design them. In principle, two essential components can be distinguished that influence the final ensemble performance - diversity and combination rule. These two aspects should be considered inseparably when designing a new method. However, they will be presented here separately.

One can say that combining models that make identical decisions makes no sense. To exploit the full potential of the classifiers' ensemble is to diversify the models at an appropriate level [271]. If one were to start building models on the same dataset, the differences among them would be minimal or none. Therefore, when designing an ensemble of classifiers, it is important to create suitable heuristics for learning models, which will allow for good separation between them [153]. There are many ways to do this, a few will be listed below:

- **Heterogeneous models** is one of the ways. It is usually assumed that an ensemble is built from models derived from the same classifier, the so-called base classifier. It is worth noting that this is not the rule. One diversification technique is to build ensembles based on different classifiers. Using models learned by various algorithms, it is called as heterogeneous ensemble [243].
- **Different training data** is one of the most well-known ensemble diversification techniques, which is to provide a variety of data used for learning. These approaches produce very diverse models. An example is bagging (bootstrap aggregating) [29], that involves dividing the learning set into smaller subsets using bootstrapping. In other words, a random sample subsets are drawn with replacement from the learning dataset.

- **Different input representations** is next diversification approach. This method involves using the same dataset for learning but inserted to classifiers in different ways. An example is the random subspace method [115]. It goes by the alternative name of feature bagging. This approach also has a bit in common with bagging. The similarity is that a draw with replacement is made, but with features. Instead of dividing the learning data into various subsets of samples, a partitioning of the feature space is performed. Specifically, this is done in such a way that each model is learned on a different randomly selected set of attributes. This allows the variance of the models to be provided, but using the full set of samples that have been designated for learning.
- **Different hyperparameters** is another approach to diversify models. This technique assumes using different settings for the hyperparameters of models. In this case, a homogeneous ensemble is created, consisting of multiple models replicated using the same algorithm. The diversity in these models is ensured at the stage of changing the learning algorithm internal settings. For example, having a k nearest neighbor classifier by manipulating the parameter k, one can easily change its characteristics and how it will classify objects.
- **Different outputs** can be used as one of the ensemble diversification techniques. Assuming that multiclass data classification is considered, very often, the problem is simplified by class decomposition. This usually comes down to the binarization, which can be done in different ways. This procedure can ensure the diversity of ensemble models. Three well-known approaches can be distinguished: OVO (One-versus-one), OVA (One-versus-all) [70] and ECOC (Error Correcting Output Codes) [61]

An equally important aspect is the measurement how much diversity is present in the classifiers' ensemble. For this purpose, certain metrics can be very useful to determine how much the models differ from each other. The strength of the model diversification can be expressed by many different techniques [156]. Some metrics such as the *disagreement measure* [232] or *double-fault measure* [96] are designed to use on pairs of classifiers. On the other hand, there are also measures to determine the diversification of the entire set of classifiers. For example, the Kohavi-Wolpert variance [134]. In general, there are many different approaches to measuring diversification. Unlikely, all of them are not listed above. However, it is important to note that there is no universal metric. Choosing the right, one depends on the problem under consideration and the effect one wants to achieve.

Another crucial aspect is a properly designed decision rule. It heavily depends on whether the built ensemble correctly classifies the data and whether the joint decision improves

the models' predictive quality [21]. A successful rule construction significantly improves the classifiers' competence by coherently combining the decisions and thereby selecting the most important values from them. Two main categories of combination rules can be distinguished - label-based and support-based. The first focuses on the use of techniques that combine answers from models that are already specifically labelled. Below is a list of selected ones:

- **Majority voting** approach is very simple in its operating principle. It is also known as hard voting. Each of the models that form the ensemble having the same right to vote. A final decision is then made by a certain voting process. Typically, when a majority of models indicate that a pattern belongs to a particular class then the pattern will be classified into that class by the ensemble. There are also some variations of this approach such as where all classifiers must agree or must be predicted by at least one more than half the number of classifiers.
- **Weighted voting** difference from majority voting in that each of the models which takes part in the decision has a certain fixed weight. These weights determine how much influence a particular model has on the final decision. This change means that no longer each of the models that form the ensemble have the same right to vote. However, it can often add a lot of value to the classification ensemble. For example, weighting can be applied that reflects the predictive ability of the models in question. Thus, having a function to evaluate how "suitable" a given model is can adjust the final ensemble decision accordingly.
- **Stacking** is another way to combine responses from models [234]. In this procedure, two types of models can be distinguished: first-level learners and second-level learner, also called meta-lerner. The idea is to train a set of first-level learners using input data and then generate a new training set from the responses of these models. However, the original class labels are kept. The newly created set of responses is used to train the second-level learner. In this simple way, a classifier is used as a direct combination rule.

The second approach focuses on using a continuous support function coming straight out of the model. This function describes the estimated support of an object belonging to a particular class. The use of such combination rules requires models with the ability to return a support function. Below is a list of selected ones:

- **Accumulated support** is also known as soft voting. It uses a majority of the models as well. However, instead of asking the models what labels they predict, the

answer sought is what support the samples get. It means what is the probability of classification object to the selected class. Robert Duin [72] proposed a division into two categories of combination rules - fixed and trained. The former focuses on using various mathematical functions such as product, sum, maximum, median rule to combine the final decision of the ensemble. The latter concentrates on more complex techniques that attempt to determine the relative contribution of each model in the final decision. This method is an interesting approach in that it can significantly change the decisions that will be made by the ensemble compared to standard majority voting.

– **Decision templates** is an approach proposed by Kuncheva et al. [155] whose idea is to create so-called decision profiles. These are matrices that try to make estimations for typical classifier responses for particular classes and predicted data samples. A critical role here is played by the similarity measure, which is the main factor influencing the final decision of the ensemble.

The one-class classifiers (Sec. 2.3) intuitively seem to be good candidates for creating the ensemble method [142]. In a one-class committee, the special decision rule should be designed to exploit individual classifiers' full potential. Tax and Duin [246] have developed a set of recommended practical tools for designing such a one-class ensemble. It is equally important to ensure that the models that make up the committee are diversified enough. One way is to divide the feature space using a clustering algorithm. Krawczyk et al. [144] have adopted an approach based on training data segmentation for *OCSVM*. A range of experiments and comparisons have shown that such decomposition significantly improves predictive quality. The ClusterSVDD [100] method works on a very similar idea. It uses k-means algorithm to create data clusters, and SVDD [247] as a one-class classifier. The DBM-EOC [172] method assumes creating clusters of one-class classifiers based on density analysis. The analysis allows identifying a subset of data that is suitable to create a cluster. Moreover, this method has a denoising capability.

2.5 Data imbalance

Imbalanced data classification is one of the issues, which is frequently explored in research. Uneven class distribution occurs in most real data sets, where the number of objects in different classes is not equal. Imbalance becomes completely different when these disparities are very significant [112]. While one set of objects describing a class dominates over the others, standard classifiers tend to overclassify as the dominant class [140]. It is usually associated with a general deterioration in classification performance. Thabtah et al. [248] have shown in their extensive research the real impact of imbalanced

data on the classification quality of selected methods in an experimental way. Usually, most of the works consider imbalanced data for the binary classification task, but this problem is also associated with multiclass data. Nevertheless, in the multiclass task, the relations among classes are not so obvious, i.e., a given class could be a majority class for one class, but it could play a minority class role for others [81]. Although, this dissertation focuses on the binary classification task, but it is worth mentioning that many approaches employ methods for a binary problem to solve multi-class ones, e.g., using smart decomposition [139].

Data imbalance can have varying degrees of severity. The *Imbalance Ratio* (IR) [93] is used to express the class imbalance intensity. This measure determines the ratio of majority class objects to minority class objects. In addition to the variation in imbalance ratio, different types of minority data distribution can be distinguished in relation to the majority class. Stefanowski and Napierala [191] proposed some division of the imbalanced data samples. According to their approach, one can determine four types of minority class instances:

– **Safe** samples that are in very uniform areas, surrounded only by minority class examples. Their location should make them much easier to predict and very valuable when learning the model (Fig. 2.7). For this reason, this type of instances can be classified as safe examples, the others belong to the three categories of unsafe instances.

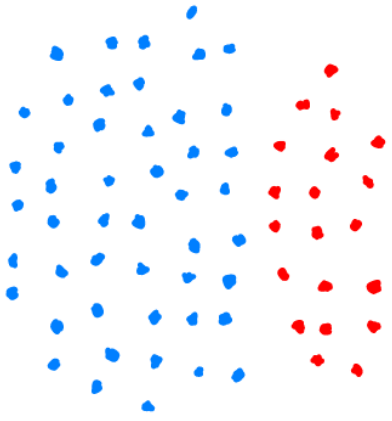


Figure 2.7: *Safe*

– **Borderline** samples that are in areas near the decision boundary, these are placed between sets of objects belonging to different classes (Fig. 2.8). Instances of different overlapping due to mixing and the difficulty of defining a clear dividing line can be included in this category. As well as those lying close to the boundary, which can be determined to a certain degree, but where there is a certain but lesser risk of incorrect classification.

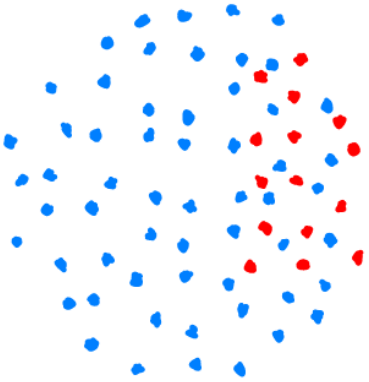


Figure 2.8: *Borderline*

– **Rare** they are represented by pairs or triples of objects belonging to the minority class placed inside the areas of the majority class. This type of samples can be treated like noise objects. However, their number is greater than one and they cannot be ignored. (Fig. 2.9). These samples have a significant impact on the predictive quality in imbalanced data problems.

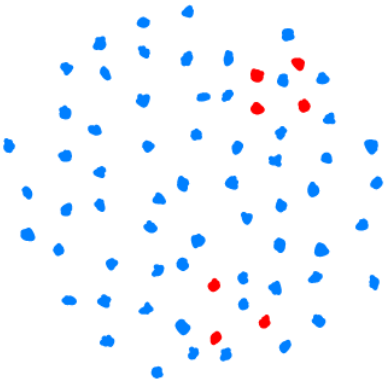


Figure 2.9: *Rare*

– **Outliers** are samples of a minority class that can easily be confused with the noise samples. They are actually individual minority class objects that can lie within the

areas of other classes (Fig. 2.10). If one were to refer to nonminority class samples, they would be assumed as noise. Because of the high value that minority objects bring to imbalanced problems, they are an essential component that cannot be ignored.

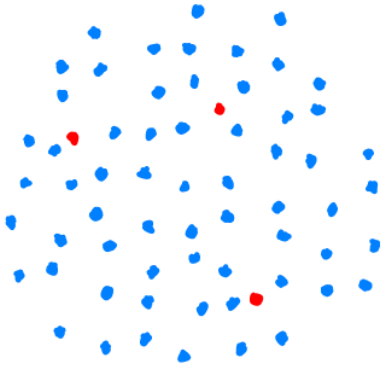


Figure 2.10: *Outliers*

2.5.1 Metrics

Class imbalance also has a very negative effect on any training process that is guided by accuracy or uniform loss functions [71]. Assuming equal importance of all training instances favors the majority class and leads to the best accuracy. Unfortunately, a bias towards the majority class will impair the generalization capabilities over the minority class, thus leading to an inefficient learning system. Therefore, *accuracy* is not appropriate metric for imbalanced data classification [40]. One of the challenges in imbalanced data classification is choosing the right set of metrics. Metrics allow to unambiguously determine the quality of the performed classification, which is expressed by a numerical value [102]. However, the metrics have very different characteristics, and it is crucial to select them properly for the classification problem.

The confusion matrix presented in Tab. 2.1 is used to define metrics mentioned in [236, 270] for binary classification. Actual values is a real class in the data and predicted values is the decision from the classifier. There are two classes positive and negative, thus there are four possibilities of counting examples belonging to specific categories. True positive (TP) and True negative (TN) stand for number of correctly classify objects into the class 1 or 0 respectively. False positive (FP) and False negative (FN) count misclassified examples. Using these calculations, it is possible to determine the confusion matrix:

Table 2.1: *Confusion matrix for binary problem*

		True	
		Positive (1)	Negative (0)
Predicted	Positive (1)	True positive TP	False positive FP
	Negative (0)	False negative FP	True negative TN

Typically, the performance of classification methods is presented using metrics rather than a confusion matrix. Below is a list of metrics along with mathematical formulas describing how to calculate them:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.12}$$

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

$$Specificity = \frac{TN}{TN + FP} \tag{2.14}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.15}$$

It is possible to measure the effectiveness of a method for multiclass classification, but these equations are slightly different and it can be done in two ways: macro-averaging or micro-averaging. In the first technique, classes are treated evenly in contrast to the second, larger classes are more important [236]. Not all metrics are appropriate for classifying imbalanced data [189], some of them such as accuracy prefers the majority class rather than the minority and it is not a desired outcome.

To illustrate this, let assume that the classification problem under consideration has a data distribution equal 10% of minority class samples and 90% majority class samples. The most commonly used metric to compare different methods is accuracy. One can also assume that to solve such a problem one would use a very strange classification model, a model that always indicates the majority class as the prediction of any sample. It is worth noting at this point, that this model, despite its large disability, will get the accuracy of 90%. This leads to a scenario where a bad predictive model that always

returns the same answer can achieve a high accuracy score. This is a very undesirable situation when evaluating a method for imbalanced data classification. To avoid such confusion, it is useful to pay attention to metrics designed to assess classification in relation to minority class predictive performance. Another approach is to use aggregate metrics, [19], which bind the classification quality of the majority class samples as well as the minority class samples:

$$F_\beta = (1+\beta^2)\frac{Precision \times Recall}{(\beta^2 \times Precision) + Recall} \tag{2.16}$$

$$BAC = \frac{Recall + Specificity}{2} \tag{2.17}$$

$$Gmean = \sqrt{Recall \times Precision} \tag{2.18}$$

$$Gmean_s = \sqrt{Recall \times Specificity} \tag{2.19}$$

$$FPR = 1 - Specificity = \frac{FP}{TN + FP} \tag{2.20}$$

$$FNR = 1 - Recall = \frac{FN}{TP + FN} \tag{2.21}$$

In addition, it is worth noting specifically that the $F_1 score$, is a very special case of the F_β (Eq. 2.16), whose parameter value β equals 1. It should be made clear that the problem of imbalanced data classification evaluation is much more complicated. Eliminating the basic disturbance in accuracy metrics caused by imbalanced data does not solve the other problems. Brzezinski et al. [40] presented visualizations of obtained results from experiments revealing the advantages and disadvantages of using selected metrics to analyze imbalanced data. These research shows that imbalanced data has a significant impact on how a metric measures classification performance. Hu and Dong [116] compare a few metrics on the premise that they contain a cost function related to the class imbalance ratio. They show that some of the metrics are appropriate for classifying imbalanced data because the misclassification's cost is higher for the minority class than for the majority class.

It is worth noting that the available metrics are not ideal for imbalanced data problems. Despite the use of various aggregate metrics such as BAC, $Gmean$ or $F_1 score$ there is

still a chance that the majority class will be favoured. Hand and Christen [106] discused that using the F_β for data imbalance should involve an appropriate adjustment of the beta parameter. This parameter adjusts how much one component (recall or specificity) affects the final score. Unfortunately, since there is no information about the loss function that determines the significance of each class one is forced to use metrics in this way.

2.6 Imbalanced data classifiers

There are two main methods for classifying imbalanced data sets that will be described below.

2.6.1 Data-level

This solution focuses on using various data preprocessing techniques. The data transformation aims at equalizing the number of training examples in each class. Garcia et al. [93] evaluate how affected is the quality with various sampling methods, which change the imbalanced data into artificially balanced data sets. There are three main approaches: oversampling, undersampling, and hybrid methods, which combine both of them.

Undersampling tries to reduce the size of the majority class. The simplest example is *Random Undersampling*, which randomly drops samples from the majority class. Unfortunately, unguided undersampling may remove important objects from the majority class, leading to incorrect decision boundaries. Most of the guided undersampling approaches employ various types of prototype selection algorithms [91]. Among them, is a popular one based on the nearest neighbors algorithm the *Condensed Nearest Neighbors* method [108]. This method discards samples from the majority class, depending on the proximity of the neighborhood. The method uses the *1-Nearest Neighbors* rule to attractively decide if the sample should be removed. *Near Miss* algorithm uses 3 different types of heuristic rules based on *Nearest Neighbors* algorithm. An interesting idea is to resample data using centroid clustering algorithms [288]. It clusters the majority of data using the selected algorithm, and the number of centroids is equal to the number of minority class objects. Next, the created centroids are new data from the majority class. This solution is an approach based on prototype generation.

Oversampling mainly focuses on generating artificial minority class samples. Since for data with a very small number of objects undersampling can lead to a significant degradation of the problem representation. The simplest and unguided approach is *Random*

Oversampling. It generates new minority class instances by duplicating randomly chosen existing objects. One of the most popular method is the SMOTE [48], which generates new samples through real minority class objects. However, one of the critical SMOTE's drawbacks is that it assumes the minority class clusters' homogeneity. It ignores the majority class instances in the neighborhood when it generates new synthetic instances. Napierala and Stefanowski [190] reported that the minority class might form small disjointed clusters, which could cause the SMOTE to increase the class overlapping and increase the difficulty of the classification task. Many SMOTE varieties overcome this drawback and extend this idea to new approaches [80]. One of the best-known version is *Borderline SMOTE* [105], which focuses on generating objects close to decision boundaries. *Safe-level SMOTE* [41] and *LN-SMOTE* [178] try to reduce the risk of generating synthetic instances inside the majority class region, while *ADASYN* [111] prioritizes the difficult instances. The SWIM [231] uses the Mahalanobis distance, through which it determines the best position of the new minority class instances, taking into account the samples from both classes. Koziarski et al. proposed *Radial-Based Oversampling* (RBO) [136] that employs potential estimation to generate new minority objects using radial basis functions. *Combined Cleaning and Resampling* (CCR) [137] method employs cleaning the decision border around minority objects and guided synthetic oversampling. It should be kept in mind that oversampling and undersampling approaches can perform in different ways. The selection of the right algorithm is crucial for good predictive performance.

Combine methods for imbalanced data processing integrate oversampling and undersampling. One exmaple is the *SMOTE-ENN* [16]. This approach generates new minority class objects with SMOTE and then cleans the data using Edited Nearest Neighbors [267]. Batista et al. [17] proposed also *SMOTETomek* method. It also uses SMOTE to create new samples but Tomek Links method to clean the data. The last category is to use ensemble balanced methods to process data. Koziarski tried to use the advantages of under- and oversampling at the same time proposing hybrid data preprocessing called *Combined Synthetic Oversampling and Undersampling Technique* (CSMOUTE). It integrates SMOTE oversampling with SMUTE undersampling [138]. Liu et al. [174] proposed two methods. *Balance Cascade* creates an ensemble of balanced sets by attractively under-sampling the imbalanced data set using an estimator, while *Easy Ensemble* also creates an ensemble of data set, but using randomly undersampling the original set of data.

2.6.2 Algorithm-level

These methods use appropriate techniques to enhance the importance of the minority class to avoid being dominated by the majority class. One of the most well-known approaches is cost-sensitive classification. They are intended to apply a cost to the classifier when a misclassification occurs. The imbalanced data is characterized by an uneven class distribution. This inequality is clearly reflected in the incurred misclassification cost [6]. For problems with strong imbalance, most often the minority class has a much higher cost. Cost-sensitive methods aim to change the importance of minority class objects during learning [295]. For example, the classifier allocates a much higher cost to false negatives in comparison to false positives, thereby highlighting any misclassified or correctly classified positive – minority class. Some transformations using cost-sensitive matrices improve the classification of imbalanced data [145].

Another cost-sensitive learning technique applied to the imbalanced dataset is to tune the decision threshold in regular machine learning algorithms. The selection of an appropriate threshold is an effective factor that affects the performance of learning algorithms. Domingos proposed *MetaCost* [63] procedure, which can convert a regular classifier to the cost-sensitive method. Other algorithms are usually based on: k nearest neighbors classifier distance modifications [300], bayesian rule-based classifier [192], or *SVM* kernel modifications [280]. Ksieniewicz [149] developed a method that employed some modifications to support domain decision boundaries.

Cost-sensitive boosting ensemble methods usually apply the boosting strategy to minimize the cost. Many of them are implementing the AdaBoost as a baseline and only some parts are changed, such as the weight update rule to provide an equal treatment of classes because AdaBoost is based on the accuracy so it is biased toward the majority class. Fan et al. proposed a modification of *AdaBoost* – *AdaCost*. During the boosting procedure, the training distribution is updated. The algorithm has regard to the cost of the misclassification in the weight update rule by adding a cost adjustment function. The weight is lower when the examples are cheaper and adequately higher when examples are more expensive. The focus of the weight updating rule is on the costly examples because weights are also increased when the system classifies incorrectly, weights are decreased otherwise [78]. *CSB* is a family of method consist of *CSB0*, *CSB1*, *CSB2* suggested by Ting. All of them modify *AdaBoost* by replacing the weight vector with different equations. *CSB0* is the simplest method developed earlier. In the weight update rule, *CSB1* and *CSB2* use confidence-rated predictions and cost which allows to minimize high cost errors [249].

It is also worth mentioning hybrid approaches that combine the advantages of data preprocessing with algorithm-level solutions. SMOTEBoost combines SMOTE algorithm with boosting procedure [49]. Galar et al. [86] employed data preprocessing (*under-* and *oversampling*) to form classifier ensemble for imbalanced data classification task.

2.7 Data stream

Substantial amounts of data are currently being produced, which often require continuous processing and analysis to obtain useful information. Thus, data stream analysis is one of the frequently studied topics in machine learning. A data stream is a massive set of data where samples may come continuously in the form of a potentially endless data stream [85]. This poses new challenges for learning algorithms, as they must adapt to ever-growing data volume and recent data distributions. Additionally, the classifier should respond quickly and its update should also be very fast [26]. The data stream's processing enforces memory and computation complexity limitations [122]. The excessive computing time can delay new data which may lead to model outdating. It can even lead to skipping some samples. It is impossible to memorize all incoming examples. Therefore, each instance should be processed as few times as possible, preferably at most once.

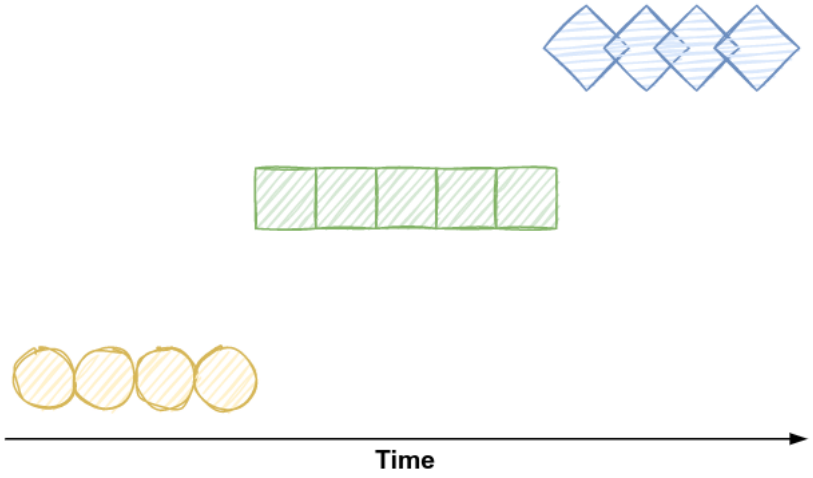


Figure 2.11: *Sudden drift*

One should mention the differences between stationary and non-stationary data stream processing. The lack of variability characterizes the stationary data streams, while the dynamic ones are associated with the phenomenon called *concept drift* [88]. This means that the distribution of feature space changes over time. These changes may occur suddenly (Fig. 2.11), incrementally (Fig. 2.12), gradually (Fig. 2.13), or recursively (Fig. 2.14). For example, suppose that a certain data stream consists of two distinct

distributions of attributes. Therefore, these are called concepts, and when a drift occurs, a transition is made from one concept to the other. However, it is reasonable to assume that these changes will continue and new or returning concepts will follow. This usually negatively affects the prediction quality of models. Depending on the speed and how these changes progress, it is possible to specify different types of concept drifts.

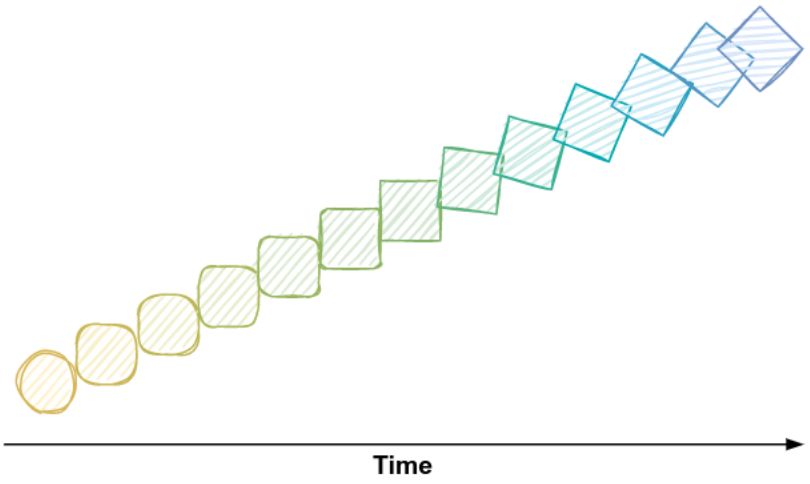


Figure 2.12: *Incremental drift*

Sudden drift (Fig. 2.11) makes changes abruptly. The transition between concepts happens very quickly without any smoothing. Incremental drift (Fig. 2.12) makes these changes extend over a larger amount of time. However, it is also necessary to consider the situation that the change will continue and a specific new concept will never be achieved. It also means that the incoming data concept changes in a very smooth manner. Gradual drift (Fig. 2.13) is a very specific type of drift. It combines the two previous approaches. The changes are abrupt in a way that there is no fluidity of change between concepts. However, the difference is that they do not move suddenly from one concept to another, but with a certain distribution, samples from new concept and ole one appear simultaneously. In the beginning there are more samples from the first concept, but over time the second concept starts to dominate until the stream switches completely to the new concept. These changes do not cause a smooth transition across the feature space as in incremental drift, but there are some jumps between the two concepts with smoothly changing intensity.

There is next type, the recursive drift (Fig. 2.14). It means that after changing the concept from one to another, eventually next drift will bring it back to origin concept. Of course, this return may be preceded by many drifts and a comeback after the transition of several different concepts, but the main idea is to return in some way to the concept that has already occurred. Recursive drift can be abrupt, incremental, or gradual, so this type of drift may be treated as a separate characteristic.

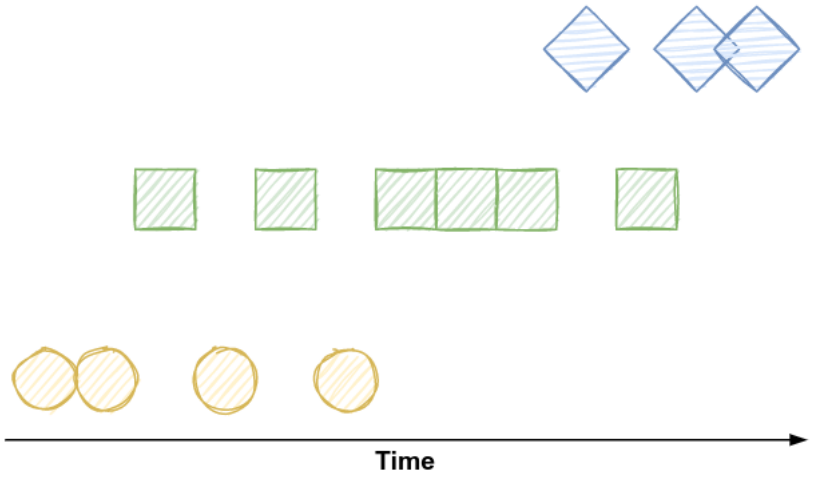


Figure 2.13: *Gradual drift*

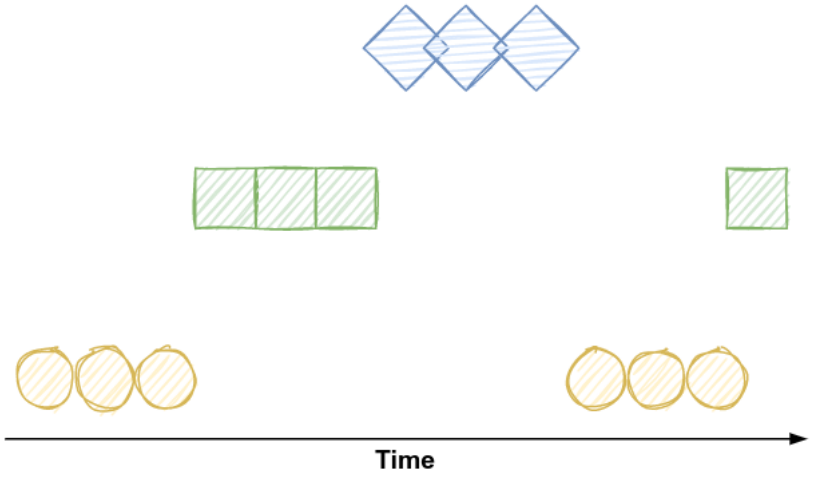


Figure 2.14: *Recursive drift*

Another obstacle that may arise when classifying data streams is the problem of data label availability. Data labeling is a process that often requires a substantial amount of time. Labels that describe how objects belong to classes can appear with a specific time delay [146]. In some cases, this delay is short. An example would be a weather prediction system whose forecast can be verified relatively quickly for the next day. Similarly, for algorithms that predict a fall or rise in the stock market, where the share prices are updated with a certain frequency, for example, every hour. It can take much longer to evaluate the credit approval system, where feedback on whether the system has evaluated the customer well can take up to several years. There are also groups of cybersecurity or fraud detection problems where the confirmation of the decision may never happen. Labels arriving with some delay and concept drift can cause that the data is out of date before use [180]. Considering this assumption, sometimes it is necessary to classify the data without full labeling during the learning process. One of the ideas for solving such

a problem is a method based on semisupervised classification. SUN method [273] uses K-modes clustering algorithm for artificial labeling and incremental decision trees for data classification with concept drift.

Correct identification of the drift type and implementing appropriate learning strategies may significantly improve classification quality. Ren et al. [209] proposed an algorithm that can deal with multiple types of concept drift. It is also important that the proposed classifier should properly react to the *concept drift*, i.e., when the drift appears, it should:

- restore a stable predictive performance as quickly as possible,
- minimize the quality deterioration after drift.

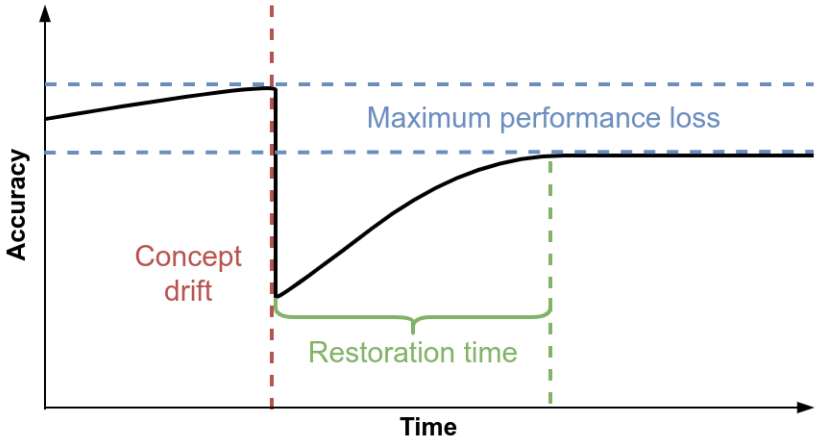


Figure 2.15: *The restoration time and maximum performance loss metrics on exemplary accuracy and time plot*

There are some works which propose how to assess the reaction of the classifier to the *concept drift* appearance. Shaker and Hüllermeier [230] proposed the two metrics: *restoration time* and *maximum performance loss* (Fig. 2.15). Initially, the proposed measures based on *Accuracy*, but any predictive performance measures (as $Gmean_s$ or F_1score) could be also used. The restoration time (RT) is defined as:

$$RT = \frac{t_2 - t_1}{T} \in [0, 1] \tag{2.22}$$

where t_1 is a chunk number for which model learning curve drops below 95% of the achieved predictive performance measure, t_2 is a chunk number for which model learning curve restores to 95% of achieved predictive performance measure, and T is the total number of chunks in a single stream. For each drift appearance, the *maximum performance loss* (MPL) is defined as:

$$MPL = \frac{PPM(t) - PPM_{12}(t)}{PPM(t)} \tag{2.23}$$

where $PPM(t) = min\{PPM_1(t), PPM_2(t)\}$ is the lower predictive performance measure value of two learning curves surrounding the drift area, and $PPM_{12}(t)$ is the lowest predictive performance measure achieved in a drift area. Apart from the above metrics, one can also calculate a mean predictive performance measure for all models. Gathered values are compared using the Wilcoxon sign-ranked test.

2.7.1 Drift detectors

There are various approaches to dealing with non-stationary data streams. One common way to deal with concept drifts is to employ drift detectors to analyze the flowing data and indicate when it occurs. Then when a drift is detected depending on the proposed algorithm, the model is updated. Barros and Santos [13] made a very interesting experimental analysis of selected methods for drift detection. Their study helps to indicate that there is really no single best method for drift detection because it depends on what type of data is present or the type of drift. An interesting observation is that the methods that are most often cited or popular do not necessarily get the best results. However, one of the most valuable outcomes of their article is that it collects most of the methods into one experiment and shows the multitude of approaches that have been developed so far.

The most popular design is the **DDM** (Drift Detector Method) [87] that measure the classifier error rate with two levels of drift detection. Estimated error rate plus two deviations for warning level and estimated error rate plus three deviations for drift detection. When DDM reaches the warning level, it begins to remember incoming samples. Lowering the error rate is treated as a false alarm, but increasing the error rate is treated as a concept drift that has occurred since the warning level. Previously collected samples are used to learn a new model, in case of false alarm samples are forgotten.

EDDM (Early Drift Detection) [10] is the modification of DDM. The method was developed to improve detection in the presence of gradual drift, where the distance between two error classifications is used instead of just considering the number of errors. **RDDM** (Reactive drift detection method) [12] is another variation of DDM method, but with a mechanism to periodically decrease the number of instances of a very long and stable concept, which causes DDM to decrease sensitivity for concept drifts.

ADWIN (Adaptive Windowing Algorithm) [23] is the drift detection method that compares two non-fixed size windows of incoming examples. Drift is detected when the

difference in averages for these windows is higher than the specified threshold. **STEPD** method [194] for drift detection uses statistical testing based on proposed formula. It depends on comparing accuracy from old samples and last one. Ross et al. [216] proposed a **EWMA** method which is a different way for drift detection using an exponentially weighted moving average. This idea of charts was originally proposed by Roberts [214] for detecting an increase in the mean of a sequence of random variables.

Dries and Ruckert proposed three drift detection methods [69]. The first method focuses on a density estimation approach using a binary representation of the data. In the second method, the average margin of the linear classifier induced by the 1-normal SVM is measured. The third uses the average error rate of the linear SVM classifier.

Frias-Blanco et al. [84] proposed few similar methods **HDDM** for concept drift detection based on the Hoeffding's inequality with proposed statistical A-test and W-test to detect significant changes in the moving average. **FHDDM** (Fast Hoeffding Drift Detection Method) [201] uses stastical test to compare the maximum probability of a correct predictions and the probability of recent correct prediction. **MDDM** (McDiarmid Drift Detection Method) [202] which simmilar idea, but based on McDiarmid [181] inequality measure.

2.7.2 Online methods

There are two different manners to the data stream classification process. Online learning where algorithms process single examples appearing one by one in consecutive moments in time [261], and chunk-based where new examples are available in so-called data chunks (blocks) consisting of many learning instances [291]. The processing way determines how a method is designed and what its capabilities are.

Online methods allow to work with single samples. It is related to model learning and prediction process. This assumption causes the online methods can predict immediately without gaining more data in one block. Most often the methods are ensemble-based approaches. Important aspect is the ability to adapt to the progressive caused by the concept drift [183]. However, they require special processing and classifiers with incremental learning capabilities.

One of the important methods in online learning is the **OB** (Online Bagging) [195]. It is a special way to apply bagging for data stream in sample by sample processing. This method has been used many times in its original version or with some modifications to other online methods. Main idea is based on using on each sample the Poisson distribution with the $\lambda = 1$ to randomly select the model from the ensemble that will be

enhanced with this sample. Bifet et al. [25] proposed an modification called **LB** (Leveraging Bagging) which uses randomization on the weights of samples from the data stream for ensemble accuracy improvement.

DDD (Diversity for Dealing with Drifts) [184] is an online data stream method which base on online bagging with few ensembles with different diversity levels. Diversity levels are ensured by changing Poisson λ values. Before drift detection there are two ensembles. One with low diversity is used for prediction and second with high diversity for drift detection. After drift detection two new low diversity and high diversity ensembles are created and the main prediction is gathered with majority voting from these four ensembles. Next, when method reaches some stability it gets back to using only two ensembles like in the initial phase.

DWM (Dynamic Weighted Majority) [135] is based on weighted majority voting ensemble. It dynamically creates and deletes models from the ensemble in response to changes in the performance of the method. DWM adds new model when ensemble makes a mistake. If the model makes a mistake, then DWM reduces its weight.

ADOB (Adaptable Diversity-based Online Boosting) [58] is also based on Online Bagging. This proposition was designed for better efficient in situations where frequent and abrupt concept drift appears. The method performs a more efficient allocation of instances between classifiers by controlling the diversity based on their accuracies. This is intended to recover more quickly from problems where concept drifts are frequently encountered.

BOLE (Boosting-like Online Learning Ensemble) [14] is a method based on a modification to ADOB idea. Three major changes were made. Firstly, the rule which keeps experts from voting has been slightly modified so that if the error exceeds 50%, the models are still not considered for the final decision, but they have the possibility to process the data. Secondly, the way weights were calculated was changed, which after the first modification allowed to achieve a negative weight. Thirdly, the drift detector was replaced from ADWIN to DDM, which increased the adaptability to different problems and added the possibility of better parameterization.

On the other hand, the **AWDOB** (Accuracy Weighted Diversity-based Online Boosting) [11] method is a slightly different approach, also heavily inspired by the ADOB idea. The main modification that has been introduced is a completely new scheme for determining weights for experts. This scheme focuses on using the accuracy of the current expert combined with the sums of correctly and incorrectly classified instances of all other experts.

ARF (Adaptive Random Forests) [97] algorithm is an approach for data stream learning using Random Forests. Leveraging bagging with Poisson binomial distribution is performed on the data. Base models use an adaptation of the Hoeffding tree classifier. The ARF has a two-stage drift detection mechanism. When a drift is detected at the warning level, new trees are built in the background, which do not affect the prediction of the method. Then, when the drift is detected, the trees are replaced with new ones and the old ones are removed.

Brzeziński and Stefanowski [37] tried to integrate weighting mechanisms and periodical models evaluations in online learning, which is most often used in chunk-based learning. The **OAUE** (Online Accuracy Updated Ensemble) was proposed as new incremental stream method. The ensemble trains and weights models with each data sample. The main idea is the cost-effective component weighting function, which estimates a models error based on last data samples, as sliding data window.

In the traditional approach, all models in the ensemble make the decision. Whether weighted majority votes or support accumulation they always influence the final prediction in some degree. In **AE** (Abstaining Ensemble) [141] authors introduced a dynamic abstaining strategy that can be used to extends abilities of any online ensemble learning scheme. In their idea, the authors assume that each classifier included in the ensemble has the option to abstain from voting. A certain confidence threshold is set to determine for each instance whether the classifier is suitable to make a decision.

Lakshminarayanan et. al [157] introduced a **MF** (Mondrian Forests) method of random forests based on Mondrian process idea [219]. It can be trained incrementally and used for streaming data in online manner. They showed that MF significantly outperforms selected online random forests in terms of training time as well as number of training instances required to meet the target accuracy result.

DELM (Dynamic Extreme Learning Machine) [279] for data stream classification is a method that uses the online learning manner to train Extreme Learning Machine [117] as basic classifier with a double hidden layer structure. Proposed method uses two levels of concept drift detection. The first level, called alert level, causes that new hidden layer nodes are added to the ELM in order to improve the generalization capability. When the next level is reached the actual ELM model is removed and a new one is built so that it learns completely from the new concept.

2.7.3 Chunk-based methods

Another branch of algorithms for classifying non-stationary data streams are methods based on processing data as data chunks. Such a block of data contains some subset of samples. This requires a certain accumulation of data, since real problems generate data instance by instance. However, this kind of processing allows for using classifiers not adapted to streams, where each model is trained on a new data chunk and added to the ensemble.

Most often, ensemble methods for classifying non-stationary data streams have a built-in mechanism for concept drift detection, which allows to react appropriately when it occurs. Barros and Santos [57] performed an experimental comparison of selected combinations of different ensembles with drift detectors. The analysis carried out did not allow the authors to identify one best method. The predictive performance was dependent on the considered data stream. However, they article shows the immensity of ensemble approaches in data stream classification problem.

One of the best known approaches for data stream classification is **Learn++** [204]. This method uses each data chunk to build a new model that is included in the classifier committee and then combines their outputs using majority voting. In the original proposition, multilayer perceptron classifiers were used. The idea eventually became fundamental to most chunk-based methods designed to classify non-stationary data streams.

Learn++NSE (Learn++ for Non Stationary Environments) [73] method based on the Learn++ idea and trains one new classifier on each data chunk. The main assumption of the authors of the method is that there is no certain fixed size of the classifier committee. Instead, weights are computed, which determine the level of impact that a particular model has on the final decision. These weights can be zero, which means that a given model does not affect the decision. On the one hand, this is an ideal method that allows great performance for data streams with recurrent concept drift. However, on the other hand, the main assumption leads to the problem where an ensemble of classifiers can be built from an infinitely large number of models.

SEA (Streaming Ensemble Algorithm) [237] also uses a learning approach which is similar to Learn++ . A certain maximum pool of classifiers is set to form the ensemble. When this size is reached, the prediction quality for all models is determined using the newest data chunk. The one with the worst score is removed. In its place, a new model trained on the most recent data chunk is inserted.

AWE (Accuracy Weighted Ensemble) [258] works in a very similar way to SEA. The main difference is that the models that build the ensemble of classifiers are continuously

evaluated on the latest data chunk. An variant of Mean Square Error is used for this measure and cross-validation of data. Pruning is also applied, which removes models performed below a certain threshold and the worst one when the maximum ensemble size is reached.

AUE (Accuracy Updated Ensemble) [36] is an AWE extension which has some innovative approaches to the problem. The authors assume that models built on a single data chunk may have poor predictive ability due to a limited pool of patterns. For this reason, an approach is proposed that some of the models are updated using a part of the samples from new data chunks. Furthermore, a way of weighting models with nonlinear error functions is introduced, where the highest weight is given to the most recent model that potentially has the best ability to classify the always changing data stream.

The AUE method was extended to **AUE2** (Accuracy Updated Ensemble 2) [38] where some ideas from AWE were abandoned. A new way of weighting the models was developed that eventually removed the use of cross-validation that appeared in the AWE method. Additionally, in this approach, all models undergo knowledge updates based on samples from new data chunks.

KUE (Kappa Updated Ensemble) [44] is a method similar to AUE but with few essential differences. Each model is built on a new data chunk, and then some updates of these models are done. New models are trained using a different subset of features and updated with instances from new chunks, but selected according to a Poisson distribution. The models are also dynamically weighted, but in contrast to AUE, the competence of a given model is determined using the Kappa statistic.

WAE (Weighted Aging Ensemble) [269] is a next chunk-based ensemble that focuses on the idea of rebuilding models during the learning process on streaming data. The method modifies the composition of the ensemble depending on the chosen diversity measure – Generalized Diversity [198]. The decision rule is weighted majority voting and the weight of each classifier depends on its accuracy and age.

ABE (Adaptive Boosting Ensemble) [53] is another proposition for non-stationary data streams with the concept drift detector mechanism. This approach is mainly based on the use of boosting to create ensembles based on simple decision tree classifiers. Despite the innovative approach, they retained the fundamental idea of dynamically assigning weights to samples, but on the other hand, the requirement of multiple data processing was excluded. In addition, a drift detector based on statistical analysis of the classification quality of the whole ensemble has been recorded.

IBS (Iterative boosting-based ensemble) [126] is a method with batch-incremental learning strategy also based on the boosting approach. The method updates models over time

and adjusts itself to data with new concepts. However, the number of models that will be added to the committee in a given data chunk is determined based on the accuracy rate of the ensemble.

BWE (Batch Weighted Ensemble) is a method designed to handle a data stream with sudden or gradual concept drift occurrence. This is a very simple approach that achieves fairly good results. The whole idea is based on the approach similar to Learn++, but using a proprietary Batch Drift Detection Detection Method (BDDM). This detector uses linear regression to analyze the accumulated classification accuracy of the classifier ensemble. The combination of iterative ensemble changes building new models on successive data chunks together with a drift detector is a very promising direction to deal with different types of drifts.

2.8 Imbalanced data stream classification

As was mentioned above, an important problem connected with data stream classification is that data distribution may change over time. Another often issue is the imbalanced data classification. Combining these two data difficulties is an additional challenge [35]. Usually, the proposed solutions focus on one of the mentioned difficulties. Methods that aim to classify data streams do not deal well with data imbalance. Similarly, classifying imbalanced data does not achieve satisfactory results when the *concept drift* may appear. When considering the imbalanced data stream classification, it is necessary to use solutions specially developed for this purpose. Methods based on classifier ensembles perform effectively with these data difficulties.

One can see that that the drifts present in the data further degrade the classification quality for imbalanced data. However, it is important to note that this is also related to metrics evaluation, which becomes even more difficult when considering non-stationary data streams with imbalanced distribution (Sec. 2.5.1). Brzezinski et al. [39] made an extensive research on the real impact of this type of data on various metrics. One of the most interesting findings is that in a data stream with changing class distributions, one can see that this affects the value of aggregate metrics, where the accuracy and precision remain static. This means that the changes that occur in the data are not exactly well expressed by all metrics.

It may also happen that the analyzed imbalanced data stream has a dynamic data imbalance [197]. In this circumstance, a class distribution may change over time (Fig. 2.16). The imbalance ratio drifts similar to the concept drift. This requires additional attention because a class that was a minority class initially may change into a majority class after

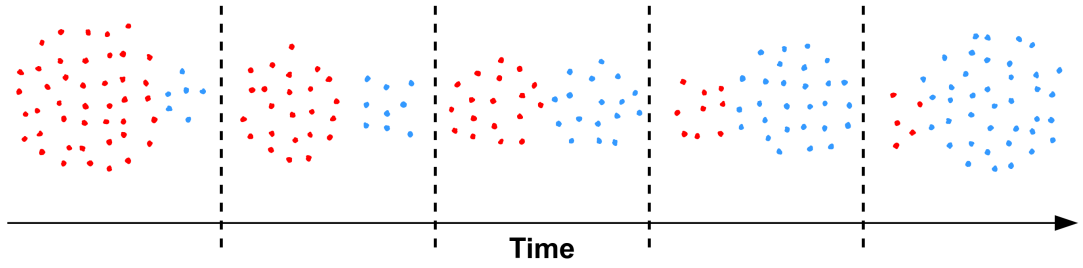


Figure 2.16: *Dynamic imbalance ratio*

some time. A very special circumstance of such dynamic imbalance may be a temporary disappearance of one particular class. For the binary task, one class's disappearance can be a severe problem to conventional classifiers. The one-class classifier (Sec. 2.3) is perfectly suited for this kind of data. There are three approaches to classifying imbalanced binary data using one-class classifiers. The first way assumes that the classifier is trained on the majority class examples only. The second way focuses on minority data only. Both of the previous solutions involve some loss of information caused by missing one of the classes. However, a much more interesting and informative solution is to classify data from both classes separately by independent classifiers. It requires designing appropriate decision rules that make it possible to combine outcomes from all classifiers.

Depending on how the data is incoming in the data stream, one may distinguish between two types of approaches. The first is the online approach, where the data stream is processed object by object [241]. This processing mode does not enforce any quantity restrictions but requires a special approach when designing the classifier [283]. A different approach is based on processing the data stream in the form of data blocks called data chunks [62]. One data chunk consists of a predetermined number of objects.

When looking for a good classification algorithm that could face many challenges, an ensemble method is often designed. The ensemble approaches have a high potential for quick adaptation [57] and the possibility to make modifications for improving prediction on difficult data [206]. The ensemble learning provides the ability to extend the competence of single classifiers [294]. One can say that such an approach has many benefits. However, to take full advantage, it requires good diversification of models [33], e.g., by training individual models based on different partitions of the training set, i.e., using vertical partitioning (different sets of feature) [250], horizontal partitioning, or using differents feature subspace using the clustering algorithm [239]. Similarly to the methods for balanced data streams, they can be divided into two subgroups - online and chunk based methods.

2.8.1 Online

In the following, selected approaches for imbalanced data streams classification will be discussed. The methods described will focus on approaches that allow for object-by-object processing of streams – online methods. This way of processing the data stream makes it difficult to build new models, especially when the data comes from an uneven distribution of classes.

EONN is the online ensemble method based on artificial neural network classifiers [94]. The main idea of this design is to build a two-layer approach. In the first layer, an cost-sensitive artificial neural networks are designated to deal with class imbalance, which means that learning samples from different classes requires different costs to be taken into account. A higher significance is assigned to errors with the minority class. This is ensured by adding weights embedded into the artificial neural networks objective function. The inspiration for authors was an work of Fontenla-Romero's cost-sensitive neural networks [82]. The second layer is an weighted majority voting ensemble to handle class imbalance and concept drifts. Authors use the Winnow-based classifier weighting scheme inspired by Kuncheva work [154].

RLSACP [95] is another method for imbalanced data stream classification based on online perceptron model. The authors were strongly inspired by the recursive least square filter in developing the new model. The changes caused by the occurrence of the drift concept are reduced by a special forgetting mechanism. The imbalance ratio is compensated by using different weighting strategies to highlight the minority class errors. The method has a high adaptability, which is due to the use of two parameters. The parameters tune the forgetting factor and the weighting of the samples, which can change dynamically during learning.

Wang, Minku, and Yao have outlined many interesting online methods for classifying imbalanced data streams. One of them is the **OOB** and **UOB** methods, both presented in the same paper [261]. Both methods consist of three separate modules. Drift detection module, imbalance detection module, and model learning module. For the imbalance detection module, they propose a certain time decay implementation, which tries to analyze the actual distribution of class labels in data stream. The next module tries to detect a drift. Due to the difficulty of imbalanced data, it focuses on each class individually. This allows the detection of drift along with an indication in which class it occurred and how much it affected the degradation of the predictive performance for that class. The last module is actually a classifier that reacts appropriately to detect concept drift or imbalanced data. OOB and UOB use an online bagging approach based on Poisson distribution. In OOB, the lambda Poisson distribution parameter is appropriately changed

to increase the chance of minority class samples. In UOB, the exact opposite happens, but for the majority class where the probability of using these samples decreases. An improved version of the OOB and UOB methods was also developed [262]. The main change was to improve the resampling strategy, making it strongly dependent on the current level of data imbalance. The second change was the proposal of weighted equivalents of these methods – **WOOB** and **WUOB**. They made the final prediction more dependent on the model that had higher quality expressed by the *Gmean*. The authors have announced one more approach that develops the previous ideas into an application for multiclass problems - **MOOB** and **MUOB** [260].

ACOG [290] is the imbalanced data stream method that focuses on using second-order information to enhance the performance of online algorithms. The authors of this algorithm have done some in-depth analysis on the use of different metrics to describe the performance of classifiers. This led them to propose their own measure based on balanced accuracy and weighted error. Then, a method was proposed for classification based on a linear classification model with an updatable predictive vector. The proposed measure was used to optimize the model. To solve the optimization problem, authors use an cost-sensitive online gradient descent algorithm.

WOS-ELM is one of the methods proposed by Mirza et al. [187] that applies extreme learning machine (ELM) to classify imbalanced data streams. As the authors point out, existing approaches to classify data streams using ELM [168], did not give successful results for the imbalanced data due to building models biased towards the majority class. The proposed WOS-ELM method solves this problem by introducing weights into the learning model to perform optimization based on the *Gmean*. However, the approach has a major drawback as it was designed to use streaming data in which there is no concept drift.

ESOS-ELM [186] is another approach from the ELM family of methods for classifying imbalanced data streams, which extends the previous proposal WOS-ELM. Unlike in its predecessor, ensemble learning based models are used here. The authors create several ELM models that have varying hidden layer parameters. A special heuristic is also proposed that decides which model a given data sample will be used for, thus balancing the training data. A drift detector is also proposed to improve predictive performance on non-stationary streams.

MCOS-ELM [185] is also an approach related to the previous two, but slightly more different in concept. In this method the meta-cognition is used to self-regulate the traning process by selecting appropriate learning strategies for class imbalance and concept drift data. The proposed approach has the ability to classify binary as well as multi-class problems which is an important advantage.

CBCE [241] is method for data stream classification in online processing. This method relies on the idea of using separate classifiers for each class of data. Each of these models is dynamically updated according to the influx of new instances. This approach allows the method to react quickly to changes in the data streams. In addition, the authors proposed a new method to handle the dynamically evolving imbalance problem caused by drifts that affect the class distribution. The presented method assumes that each classifier has a inbuilt mechanism for dynamic data balancing, where samples from majority classes are discarded when the number of samples significantly deviates from the size of minority class samples. The removal of samples is performed with a certain probability distribution that additionally relies on a decay over time.

2.8.2 Chunk-based

Now it will be discussed methods for classifying imbalanced data streams that work in a chunk-based manner, where data from a data stream is extracted as blocks containing subsets of patterns. Processing the stream in this way is easier than online and allows for a variety of preprocessing techniques that are known in applications to static imbalanced datasets.

Ditzler et al. [62] proposed two methods based on Learn++ [204] algorithm, where new models are built on a subsequent data chunks. One of them, **Learn++CDS**, is an extension of the Learn++NSE [74] method, which was originally designed to classify non-stationary data streams. The main extension focus on employing data oversampling method – SMOTE. The combination of these techniques allows an appropriate response to drifts using a weighted decision rule.

The second method, **Learn++NIE**, is a slightly more complicated approach. Firstly, a custom bagging is used to automatically balance the data chunks. This is done by applying heuristics that select samples from different classes, so that the final class distribution is balanced. This lead to the method where new models that employ the ensemble, are composed by subset of classifiers. Weights are calculated in a slightly different way, based on classification error.

SERA [50] is a method for classifying imbalanced data streams that uses minority class patterns from previous data chunks to balance the current one. The amplification of these patterns is performed in the following way. The Mahalonobis distance to the current minority class data is calculated. The resulting list of distances is sorted from highest to lowest. Then the few first samples selected for amplification are chosen according to this list in order to obtain a given imbalance ratio of the data chunk. The resulting set of patterns is used to create a model using bagging technique.

MuSeRA [52] is the approach that makes several modifications to the previous SERA method. The main change is that instead of using bagging, models are built on subsequent data chunks. This is a solution where the ensemble builds over time. In addition, the generated models have a calculated weight based on predictive performance, which is taken into account when determining the final decision of the whole ensemble of classifiers.

REA [51] is essentially another method that is a continuation of the previous two approaches. The biggest innovation of this idea is rule of selection the stored minority enhance the current data chunk. The Mahalonobis metric is discarded and instead sample selection using the KNN classifier is applied. For each retained minority sample, a label is determined using KNN, where the training data is the current data chunk. Then their probabilities of belonging to the minority class are saved and a list is created according to these values starting from those with the highest value. The number of best matching samples is then selected to get the desired imbalance ratio in the data chunk.

KMC [264] is a slightly different approach to classifying imbalanced data streams. They based their method on a very interesting data undersampling technique - the K-means algorithm. Before explaining more precisely the essence of this approach, it is worth noting that most methods for undersampling data focus on a using prototype selection techniques. In other words, these methods try to select the best subset of patterns that will produce a more balanced set, simultaneously obtaining a good predictive performance. In this case, the samples are not selected. However, they are replaced by new ones, but with smaller size. The whole process goes in such a way that the K-means algorithm determines as many centroids as there are samples of the minority class. Then these centroids replace the objects of the majority class. A new model is trained this new data. This process is repeated with each subsequent data chunk. Additionally, the models are weighted using the AUC.

OUSE [89] is a method that uses several interesting approaches to combat data imbalance and concept drift. First, each data chunk undergoes a twofold balancing. Initially, a minority class enhancement is performed using samples from the previous data chunk. Then some undersampling is done to further reduce the differences that remain between the class distributions. This is done in such a way that the majority class data is divided into enough parts to get an equal amount of patterns with the minority class. Later, each of these sets, together with a minority class objects, are used to learn a new model. In this way, both a clever undersampling of the data and also some bagging is done to create an ensemble of classifiers.

Ren et al. proposed two very simmilar methods – **GRE** (Gradual Resampling Ensemble) [210] and **SRE** (Selection-based Resampling Ensemble) [211], to classify imbalanced

data streams in online and chunk-based processing. The presented methods focus on using selectively resampling mechanism. This is a technique that, as in previous algorithms, stores data only from the minority class to enhance the newest data chunk. This procedure is a bit more complicated and requires special attention. First, the data from each class is divided into clusters. Then the Mahalanobis distances among these clusters and the stored patterns are determined. Based on the obtained values, a list is created according to which new samples are selected to strengthen the minority class of the actual data chunk. In addition, the authors proposed a mechanism for ensemble pruning, where the worst performing models are removed, and a method for determining model weights based on Mean Square Error.

MDE (Minority Driven Ensemble) is an approach proposed by Zyblewski et al. [301]. The method focuses on a simple, but successful idea for the selection of classifiers. The ensemble is built by incrementally adding new models, which are trained on subsequent data chunks. When at least one of the models forming the ensemble indicates that the object belongs to a minority class, then the ensemble makes a final decision based only on that model. This technique effectively improves the model's ability to classify minority data. This approach has no drift detector, but has a good ability to self-adapt to a changing data distribution.

DUE (Dynamic Updated Ensemble) [167] is an approach to classify imbalanced data streams that uses model updating and bagging techniques. One of the main components of this algorithm is the procedure for building new models. Each data chunk contributes to the creation of a set of models. This is done using controlled bagging, where samples are drawn from each class in sufficient numbers to provide a balanced training set for a single model. In situations where there are too few minority class objects to perform such a draw well, the SMOTE algorithm is used, which further samples the new patterns. Then the evaluation and weighting of all models is performed. This is done using a proposed formula consisting of a combination of Mean Square Error and Hellinger Distance of feature spaces between models. Then those models that get the weakest weight and are above the assumed ensemble dimension are removed. The ensemble decision is made by weighted majority voting.

TSCS (Two-Stage Cost-Sensitive) [240] ensemble is a method for classifying imbalanced data streams that uses a two-stage cost-sensitive learning scheme. In the first stage, feature selection is performed using the Cost-Sensitive Principal Component Analysis (CSPCA) algorithm. It selects features and balances the data using sample weighting. Different weights for minority and majority class samples can reduce the dominant role of majority instances in feature selection. In the next phase, a set of weighted models is created using cost-sensitive modifications. This approach allows dual application of

data balancing without changing the data distribution. Additionally, the method has an "adaptive window change detector" built in to indicate when is the best time to build a new candidate classifier and adapt to any drift quickly.

2.9 Data noises

One of the equally important problems that arise in data is noise. It is some indispensable part of real world data. Most studies should take into account the existence of the certain disturbance in the form of noise, which makes the task of classification even more difficult. In the following section, it will be discussed two types of noise that can be distinguished – label noise and attribute noise. In addition, ways to deal with this type of data will be presented.

2.9.1 Label noise

The quality of labeling has a significant impact on the predictive quality of the classifiers. The data labeling process is not error-free. It can have a different character and can be viewed as noise introduced into the learning information. One may consider the noise caused by a human operator (incorrect imputation) or measurement errors when acquiring attribute values. Label noise occurs whenever an observation is assigned incorrect label [299], and can lead to the formation of contradictory learning instances, if duplicate observations are assigned to different labels [114]. The problem of incorrect labeling has been considered in the literature for years [92, 227], including a survey by Frénay and Verleysen [83]. However, relatively few papers focused on the dependency between data noise and predictive performance of imbalanced data classifiers [136]. To start discussing the nature of data noise, one should first consider where the information on the labels is obtained. Human experts give the most common labels, but should be taken into account:

- man is not infallible, e.g., considering the quality of medical diagnostics, it can be concluded that the number of errors made by human experts is noticeable [66],
- the distribution of errors committed by experts is not uniform, because labeling may be subjective,
- human experts may be biased,
- human experts are not available 24/7 and their work is costly.

Another approach is obtaining labels from non-experts as crowdsourcing. It provides a scalable and efficient way to construct labeled datasets. However, creating comprehensive label guidelines for crowd workers is often hard, even for seemingly simple concepts. Incomplete or ambiguous label guidelines can result in differing interpretations of concepts and inconsistent labels. Also data corruption can cause label noise [46], e.g. – *data poisoning* [166] and adversarial attacks [285]. The label corruption brings a predictive performance degradation of classification systems [113]. The label error distribution may have a different nature, usually dependent on error source and one may distinguish:

- a completely random label noise,
- a random label noise dependent on the true label (asymmetrical label noise),
- a label noise dependent on the true label and features.

There are several methods to combat label noise. One of the most popular is *data cleaning*, where e.g., SMOTE is applied with cleaning using the *Edited Nearest Neighbors* (ENN) [16]. This approach keeps the relatively high number of observations, and the number of mislabeled observations is relatively low, allowing to detect improperly labeled observations. Designing a *label noise-tolerant* learning classification algorithm is another approach. It assumes a model of label noise distribution and analyzes the viability of learning under this model. Angluin and Laird present an example of this approach as a *Class-conditional noise* model (CCN) [9].

Finally, the last approach is designing a *label noise-robust* classifier. Even if there is no data denoising, nor any noise is modeled, still produces a model that has a relatively good predictive performance when the learning set is slightly noisy [83].

2.9.2 Attribute noise

Most real world datasets have some error that decide about quality of the data. When the classifier gets the training data with errors, the more difficult it is to get the model because such noise distorts the data distribution, which can lead to the model's overfitting to noise. The best idea is to create a model for non-noisy data but the manual data cleaning process is ineffective, time-consuming and expensive, so it is worth using preprocessing methods to remove noisy samples, fill in missing values or revise incorrect values. However, removing noise from attribute values is more difficult and less common in literature than the case of label noise [104], because it is understood that individual features are not as important as the class. Some features may be redundant or of little use [298]. Contained noise may result from the experiment or a real situation from which

the data is collected e.g. faults in apparatus, transcription and transmission errors, and consequently noise is unpredictable [91]. Zhu and Wu [298] proposed that attribute noise is consider when errors contain:

- erroneous attribute values
- missing or do not know attribute values
- incomplete attributes or do not care values

Zhu et al. [298] add attribute noise to the training set and the test set, and they make cross-evaluations to check the classification accuracy. They show that this noise reduces the prediction performance and it may be beneficial to not discard all instances but to adjust them. Attributes may be correlated with each other, and then it is not necessary to handle the noise of all features because some of them have a greater impact on the model performance than others. Comparing these two types of noise, attribute and class noise, they claimed that class noise is more harmful than attribute noise. One of approaches that tries to combat attribute noise is using class noise filtering where class is replaced by each attribute, but it can be applied only when attributes can be predicted by other attributes, otherwise different techniques should be used, such as clustering, k nearest neighbors, association analysis.

Yu et al. proposed a learning paradigm to reduce the attribute and label noise from credit risk assessment datasets [284]. The algorithm has three stages: (1) Calculating values of information gain, gain ratio, Gini index and Sperman's correlation coefficient for each feature; (2) Two-round voting for attribute categorization into different sets; (3) Processing attribute noise containing different learning strategies for different sets obtained in the step 2, non-sparse or sparse processing, and de-noising. Yang et al. [281] proposed the method handling noise of predictive but unpredictable attributes. These attributes are useful to classification and they cannot be predicted by others features so it is crucial to dealing with the noise. The attribute noise is identified, cleaned and measured by proposed conceptual equivalence measurement. **PANDA** (Pairwise Attribute Noise Detection Algorithm) was proposed in [255]. The algorithm compares two attributes at a time, and assigns Noise Factor to rank attributes from most noisy to least noisy in order that remove the most noisy attributes and obtain cleaner dataset. **HNOML** (Hybrid noise-oriented multilabel learning) [287] is based on bi-sparsity regularization combined with label enrichment to handle both attribute and label noise in the multilabel classification.

The above chapter presents the most important issues of machine learning and research areas that will be partially addressed in this dissertation. Concluding, it is worth noting

that the classification of imbalanced data streams poses many challenges both from the side of developing classification methods as well as the research workshop, including properly conducted experimental evaluation. Moreover, it is worth keeping in mind that despite the niche nature of the classification of imbalanced data streams, it is important to remember about the existing contributions to this domain.

Chapter 3

One class classifier ensembles

This chapter focuses on presenting the methods for classifying imbalanced data streams using ensembles of one-class classifiers. Two different approaches are introduced and subjected to experimental evaluation with statistical analysis of the obtained results. The first one concentrates on using the committee with one-class classifiers trained on clustered data for each class separately. The next approach extends the previous one by proposing a new weighted combination rule along with a much more extensive experimental analysis.

3.1 One class support vector machine classifier ensemble for imbalanced data stream

The approach proposed in this section makes a very heavy attempt – using one-class models for binary imbalance data stream classification. From one side, it is adding difficulty, because such classifiers are designed mainly for tasks such as anomaly detection or novelty detection – one class problems. On the other hand, intuition suggests that an imbalanced data stream may turn out to be an ideal candidate for becoming such an approach. First of all, it is worth noting that the drift in imbalanced streams can cause dynamic changes of the imbalance ratio. In very extreme situations, this can lead to the temporary or complete disappearance of one class. Class vanishing for a standard binary classifier can be a bit of an insurmountable problem, as the lack of counter examples will not allow the classifier to keep running. It is in this situation that a one-class classifier, which only needs data from one class to build the model correctly, may find an ideal application. Following this idea, a **O**ne **C**lass support vector machine classifier **E**nsemble for **I**mbalanced data **S**tream (*OCEIS*) method has been proposed.

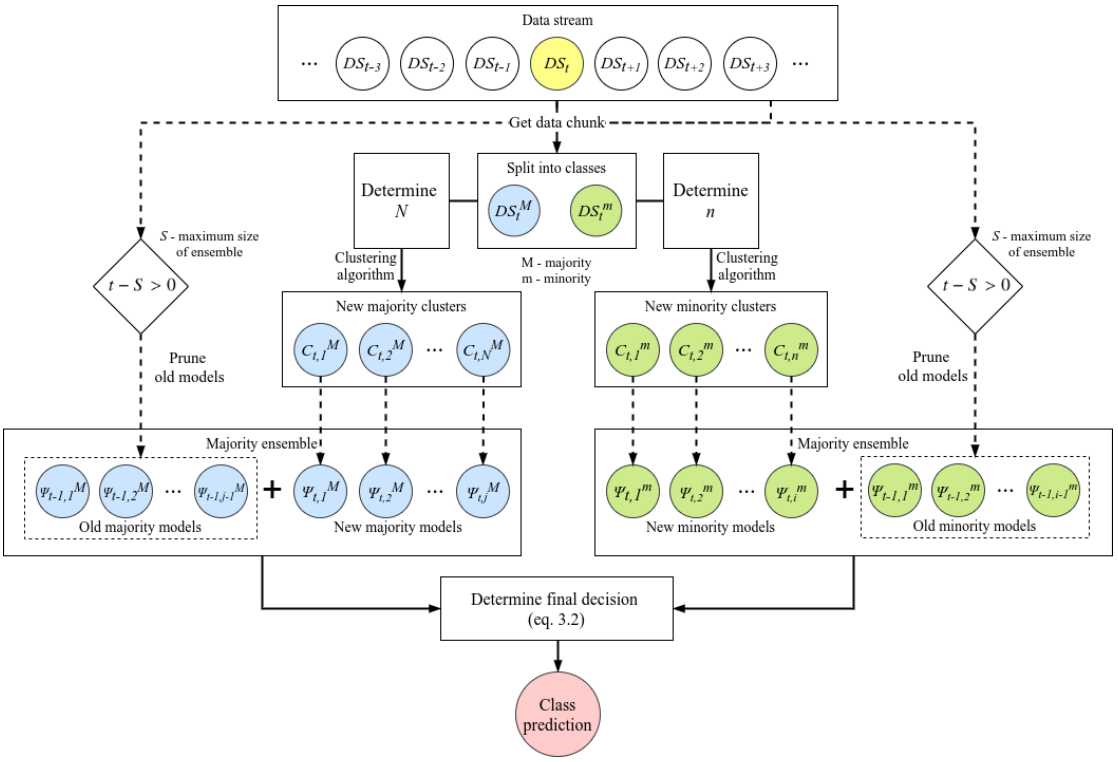
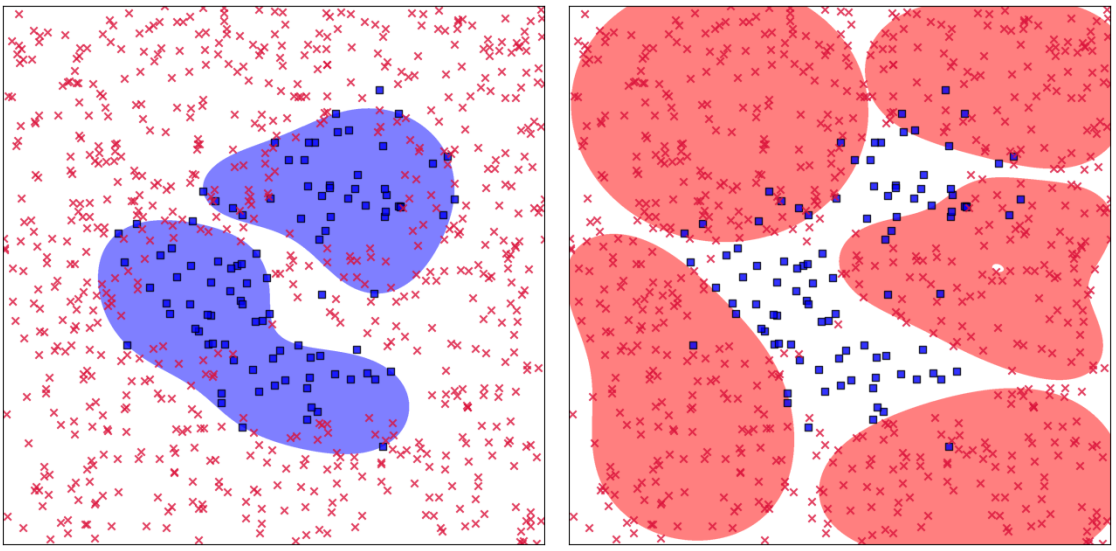


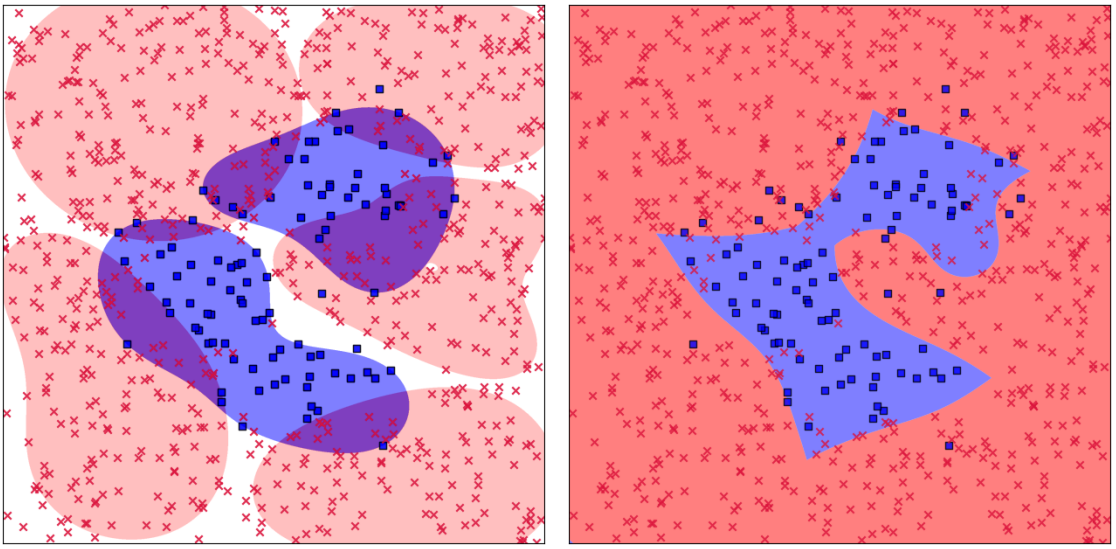
Figure 3.1: *OCEIS flow diagram*

The main core of this idea is the use of one-class support vector machines ($OCSVM$) to imbalanced binary classification tasks. Classification of imbalanced data streams is a complex problem. Usually, such tasks are solved by a classifier ensemble, which has much better ability to deal with difficult data. Similarly, in this method, an ensemble composed by many one-class classifiers has been designed. This proposition is a chunk-based data stream method. However, the deployment of ensemble learning involves ensuring that the required conditions to fully exploit the potential of multiple models. Firstly, the focus should be on proper ensemble diversification. The idea of this method is to use clustering of the learning data. Each model is trained on a different set of data. In addition, a suitable combination rule should also be designed to extract knowledge from individual models. Fig. 3.1 presents the main idea of the proposed method.

In the first step of the Alg. 1, the training data chunk (DS_t) is divided into a minority (DS_t^m) and a majority set (DS_t^M), where t is current time stamp, m means minority and M means majority. Then these sets of data are divided into clusters using the clustering algorithm (CA). Krawczyk et al. [144] indicate the importance of this idea. This decomposition of data over the feature space allows achieving less overlap of classifiers decision areas in the ensemble (Fig. 3.2). The K-means algorithm [179] is used to create clusters. The key aspect is choosing the right number of clusters. One such solution may be the use of cluster consistency metrics (CM). Silhouette Value (SV) [217] comes



(a) Minority classifiers *(b) Majority classifiers*



(c) Both classifiers *(d) Ensemble*

Figure 3.2: *Decision regions visualisation on the paw dataset from the KEEL repository[5]*

with help, which allows calculating how similar an object is to its own cluster compared to other clusters. Kaufman et al. [165] introduced the Silhouette Coefficient (SC) for the maximum value of the mean SV over the entire dataset. This metric describes how similar objects are to their cluster compared to other clusters and it is described by the Eq. 3.1:

$$SV(x) = \frac{b(x) - a(x)}{max\{a(x), b(x)\}}, \tag{3.1}$$

Where x is an object from cluster C, $a(x)$ is a measure of how much x dissimilar to

other clusters and $b(x)$ is a measure of how much x dissimilar to cluster C. Both of these metrics are calculated based on the average distances which are expressed by the Euclidean distance measure. In addition to the metric itself, some procedure is needed to search for the optimal solution. The idea for this is to explore different solutions in a predefined range.

Minority and majority data are divided into two cluster sets ($C^m_{t,k}$, $C^M_{t,k}$) with a different number of centroids – k from 1 to K, where K is an input parameter. The number of clusters for minority (n) and majority (N) is selected by the option with the highest value of CM. This process is performed for minority and majority data. Then the formed clusters are used to train new $OCSVM$ models ($\Psi^m_{t,i}$, $\Psi^M_{t,j}$). These models are included in the pool of classifier committees (Π^m, Π^M). The method is designed by default to operate on data streams. For this reason, a simple forgetting mechanism, was implemented. This allows using models trained only on data with a certain time interval. When the algorithm reaches a value $s = t$, in each iteration, the models built on the oldest chunk are removed from the ensemble.

A crucial component of any classifier ensemble is the combination rule, which makes decisions based on the predictions of the classifier ensemble. Designing a good decision rule is vital for proper operation and obtaining satisfactory classification quality. First of all, *OCEIS* uses one-class classifiers and class clustering technique, which changes the way how the ensemble works. Well-known decision making based on majority voting does not allow this kind of committee to make correct decisions [278]. The number of classifiers for individual classes may vary significantly depending on the number of clusters. In this situation, there is a considerable risk that the decision will mainly base on majority classifiers. *OCEIS* uses the original combination rule based on distance from the decision function which determines the relative distance of classifiers boundries to predicted samples. Let $f^m_{t,i}$ denotes discrimination function of classifier $\Psi^m_{t,i}$ (Eq. 2.10) and $f^M_{t,j}$ of classifier $\Psi^M_{t,j}$ respectively. The final decision of the *OCEIS* is made according to the following classification rule:

$$\Psi(x) = \begin{cases} \text{minority class} & if \max\limits_{i \in \Pi^m} \left(f^m_{t,i}(x) \right) \geq \max\limits_{j \in \Pi^M} \left(f^M_{t,j}(x) \right) \\ \text{majority class} & \text{otherwise} \end{cases} \tag{3.2}$$

Algorithm 1 OCEIS

Input:

DS – data stream
DS_t – t-th data chunk of data stream DS
s – maximum size of classifier ensemble
K – maximum number of clusters
CA – clustering algorithm, e.g. K-means [179]
CM – clusters consistency, e.g. Silhouette Coefficient metric [165]

Symbols:

DS_t^m – minority data in the t-th chunk
DS_t^M – majority data in the t-th chunk
n – number of minority clusters
N – number of majority clusters
$C_{t,k}^m$ – clusters of minority data DS_t^m
$C_{t,k}^M$ – clusters of majority data DS_t^M
$\Psi_{t,i}^m$ – $OCSVM$ model trained on $C_{t,i}^m$ cluster
$\Psi_{t,j}^M$ – $OCSVM$ model trained on $C_{t,j}^M$ cluster
Π^m – minority model set (ensemble)
Π^M – majority model set (ensemble)

Output:

Π^t – final ensemble for t-th data chunk

for $t = 1, 2, \ldots$ **do**
  Split DS_t into minority (DS_t^m) and majority (DS_t^M) data
  for $k = 1, 2, \ldots, K$ **do**
    $C_{t,k}^m \leftarrow$ Create k clusters using CA on DS_t^m
    $C_{t,k}^M \leftarrow$ Create k clusters using CA on DS_t^M
  end for
  $n \leftarrow \underset{k=1,2,\ldots,K}{\operatorname{argmax}} \; CM(C_{t,k}^m)$
  for $i = 1, 2, \ldots, n$ **do**
    $\Psi_{t,i}^m \leftarrow$ Train $OCSVM$ model on $C_{t,i}^m$ cluster data
    $\Pi^m \leftarrow \Pi^m \cup \Psi_{t,i}^m$
  end for
  $N \leftarrow \underset{k=1,2,\ldots,K}{\operatorname{argmax}} \; CM(C_{t,k}^M)$
  for $j = 1, 2, \ldots, N$ **do**
    $\Psi_{t,j}^M \leftarrow$ Train $OCSVM$ model on $C_{t,j}^M$ cluster data
    $\Pi^M \leftarrow \Pi^M \cup \Psi_{t,j}^M$
  end for
  if $t > s$ **then**
    $\Pi^m \leftarrow \Pi^m \setminus \Psi_{t-s,j}^m$
    $\Pi^M \leftarrow \Pi^M \setminus \Psi_{t-s,i}^M$
  end if
end for

Computational complexity analysis

In this section, the computational complexity of the proposed method will be presented. *OCEIS* can be divided into two fragments, which makes up the computational complexity of the whole algorithm. First is the base classifier. According to the main idea of this proposition, an *OCSVM* classifier is used here, whose complexity is equal to $O(n_s^3)$ [275], where n_s is the number of objects. The second factor affecting the complexity is the method to cluster the data. Since this is a parameter for the *OCEIS* method, the complexity of this element may vary depending on choice. In this case, it can be assumed in advance that the K-means method will be used, which has a complexity equal to $O(kt_in_s)$ [123] where k is the number of clusters, and t_i is the number of iterations. The final complexity of the *OCEIS* algorithm is $O(n_s^3 + kt_in_s)$.

3.1.1 Experimental evaluation

The main purpose of this experiment was to check how well the proposed method performed in comparison to the other methods for classifying imbalanced data streams. The following research question was formulated:

RQ1: Is it possible to design a one class ensemble method with a statistically better or equal predictive performance than the selected *state-of-the-art* methods for imbalanced data streams?

Setup

Table 3.1: *Parameter setup for data stream generator*

Parameter	Value			
Number of samples	100000			
Number of chunks	200			
Chunk size	500			
Number of classes	2			
Number of features	10 (8 informative + 2 redundant)			
Number of drifts	5			
Concept drift types	sudden	incremental		
Random state	1111	2222	3333	4444
Imbalance ratio	10%	20%	30%	

All tests were carried out using 24 generated streams (Tab. 3.1 and 30 real streams (Tab. 3.2). The size of the data chunks in real data was chosen experimentally. Data chunks were processed in a *test-then-train* manner [24]. The generated data comes from

stream-learn [150] generator. The proposed method has been tested with the selected *state-of-the-art* methods:

- *REA* – Recursive ensemble approach [51]
- *KMC* – K-means clustering undersampling ensemble [264]
- *L++CDS* – Learn++CDS [62]
- *L++NIE* – Learn++NIE [62]
- *OUSE* – Over and undersampling ensemble [89]
- *MLPC* – Multi-layer perceptron classifier

Experiments were implemented in Python programming language. The SVM implementation from the *scikit-learn* framework [200] was used as the base classifier for *state-of-the-art* methods. *OCEIS* implementation and the experimental environment is available on public Github repository[1]. The experiments were evaluated using five different metrics:

- $Gmean_s$
- $F_1 score$
- *Recall*
- $Specificity$
- *Precision*

Then results obtained in this way were compared using Wilcoxon pair rank-sum tests.

Experiment 1 - State-of-the-art comparison

The obtained results of the Wilcoxon rank-sum pair statistical tests show that *OCEIS* can classify with the similar quality compared to the tested methods. There is a certain advantage of the L++CDS method over other methods for tested synthetic data streams (Fig. 3.3). In the second place it can be put L++NIE and *OCEIS*. For the OUSE and L++NIE methods, there is a noticeable tendency to classify objects of the minority class, which is manifested by the higher results in the *Recall* (TPR), but this causes a significant drop in $Specificity$ (TNR). The worst in this test was the REA method, which shows a huge beat in the direction of the majority class. The results are more

Table 3.2: *Overview of real datasets used in experimental evaluation (KEEL [5] and PROMISE Software Engineering Repository [170]), IR – Imbalance Ratio*

Dataset	IR	Samples	Features	Chunk size
abalone-17_vs_7-8-9-10	39	2338	8	250
australian	1.2	690	14	50
elecNormNew	1.4	45312	8	500
glass-0-1-2-3_vs_4-5-6	3.2	214	9	30
glass0	2.1	214	9	30
glass1	1.8	214	9	30
heart	1.2	270	13	50
jm1	5.5	2109	21	100
kc1	5.5	2109	21	100
kc2	3.9	522	21	50
kr-vs-k-three_vs_eleven	35	2935	6	250
kr-vs-k-zero-one_vs_draw	27	2901	6	250
page-blocks0	8.8	5472	10	500
pima	1.9	768	8	50
segment0	6	2308	19	100
shuttle-1vs4	14	1829	9	100
shuttle-1vsA	3.7	57999	9	500
shuttle-4-5vsA	3.8	57999	9	500
shuttle-4vsA	5.5	57999	9	500
shuttle-5vsA	17	57999	9	500
vehicle0	3.3	846	18	100
vowel0	10	988	13	50
wisconsin	1.9	683	9	100
yeast-0-2-5-6_vs_3-7-8-9	9.1	1004	8	100
yeast-0-2-5-7-9_vs_3-6-8	9.1	1004	8	100
yeast-0-3-5-9_vs_7-8	9.1	506	8	70
yeast-0-5-6-7-9_vs_4	9.4	528	8	60
yeast-2_vs_4	9.1	514	8	60
yeast1	2.5	1484	8	150
yeast3	8.1	1484	8	150

transparent for real data sets (Fig. 3.4). Despite many ties, the best performing method is *OCEIS*. The exceptions are *Recall* for OUSE and *Specificity* for REA.

[1]Repository link: https://github.com/w4k2/oceis-iccs2020

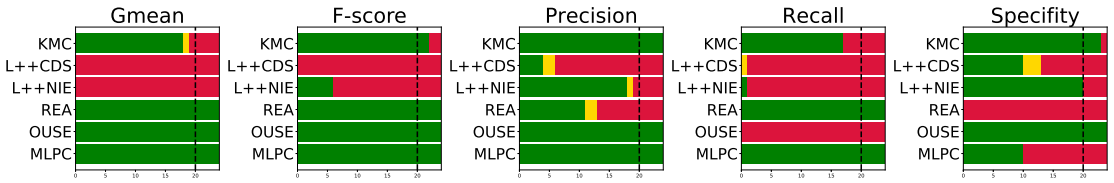


Figure 3.3: *Wilcoxon pair rank sum tests for synthetic data streams. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – lose)*

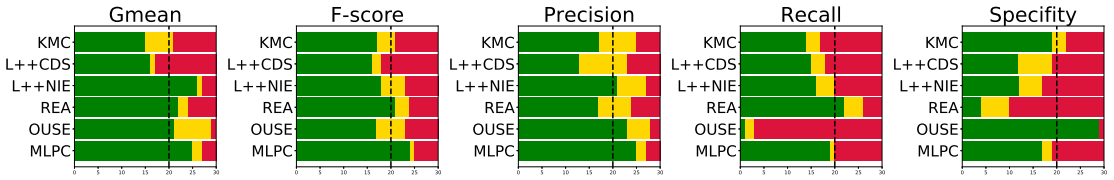


Figure 3.4: *Wilcoxon pair rank sum tests for real data streams. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – lose)*

Charts of $Gmean_s$ score over the data chunks provide some useful information about the obtained results. To get a much better readability, the data before plotting was processed using a Gaussian filter. This procedure smoothes the edges of the results, which allows getting much more information from the results. The first observation is that *OCEIS* does not degrade the quality over time for the incremental drift stream (Fig. 3.5). The negative effect of the concept drift can be seen with the KMC and REA methods, where the quality deteriorates significantly with the inflow of subsequent data chunks.

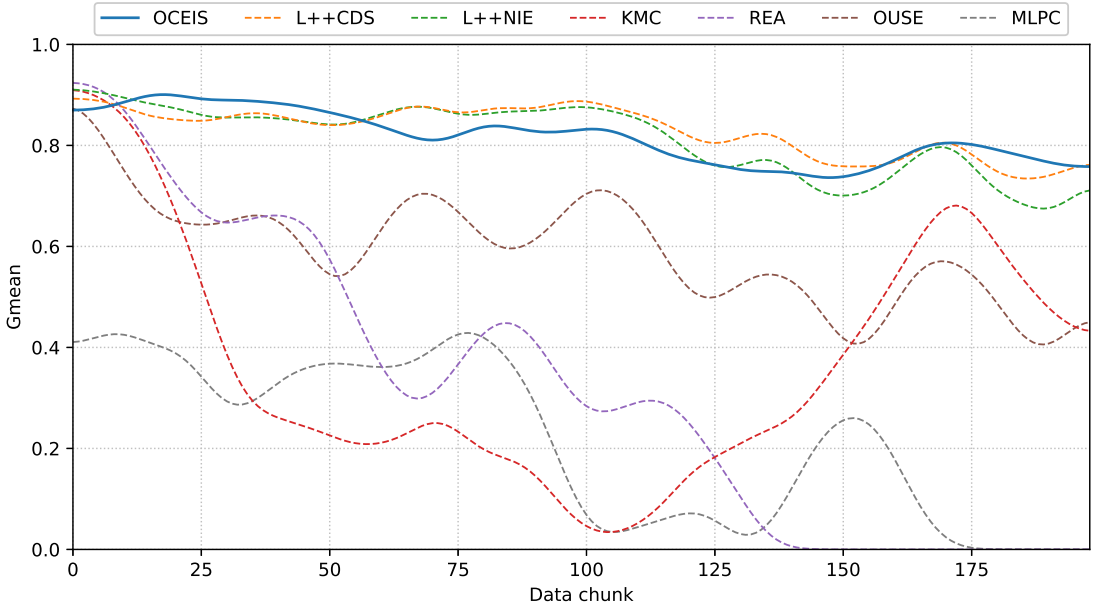


Figure 3.5: *$Gmean_s$ score over the data chunks for synthetic data with incremental drift*

In sudden *concept drift* (Fig. 3.6), a certain decrease is noticeable, which is more or less reflected on every tested method. However, L++CDS, L++NIE and *OCEIS* can quickly

rebuild this quality drop. This does not affect the overall quality of the classification significantly. Other methods perform a little bit randomly on sudden drifts. An example of the real-time shuttle-4vsA stream (Fig. 3.7) shows the clear advantage of the *OCEIS* method over the other tested methods. A similar observation can be seen in other figures for real streams.

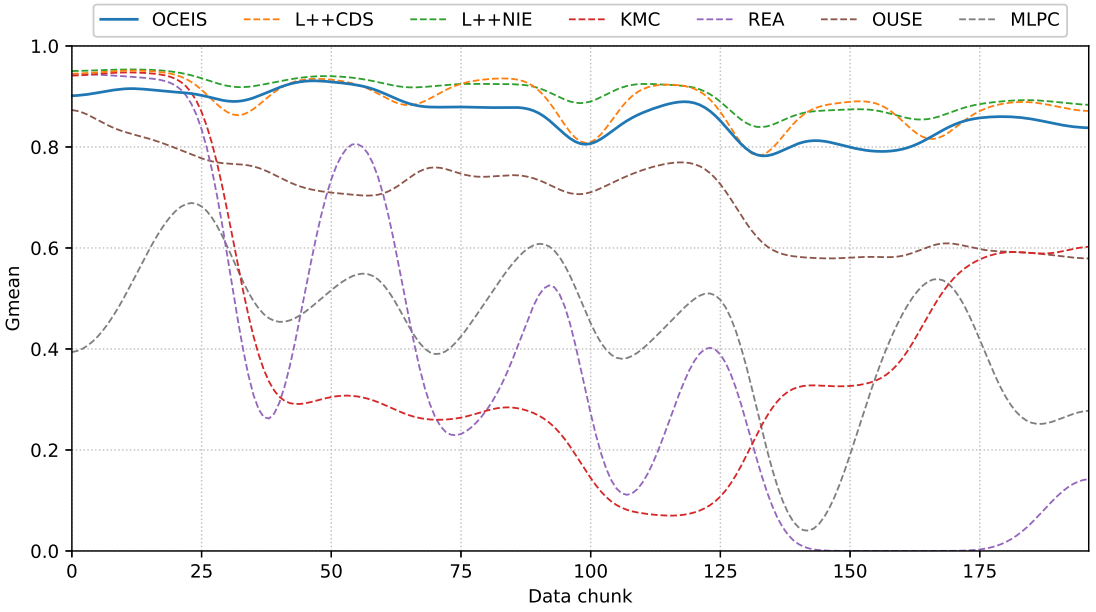


Figure 3.6: *$Gmean_s$ score over the data chunks for synthetic data with sudden drift*

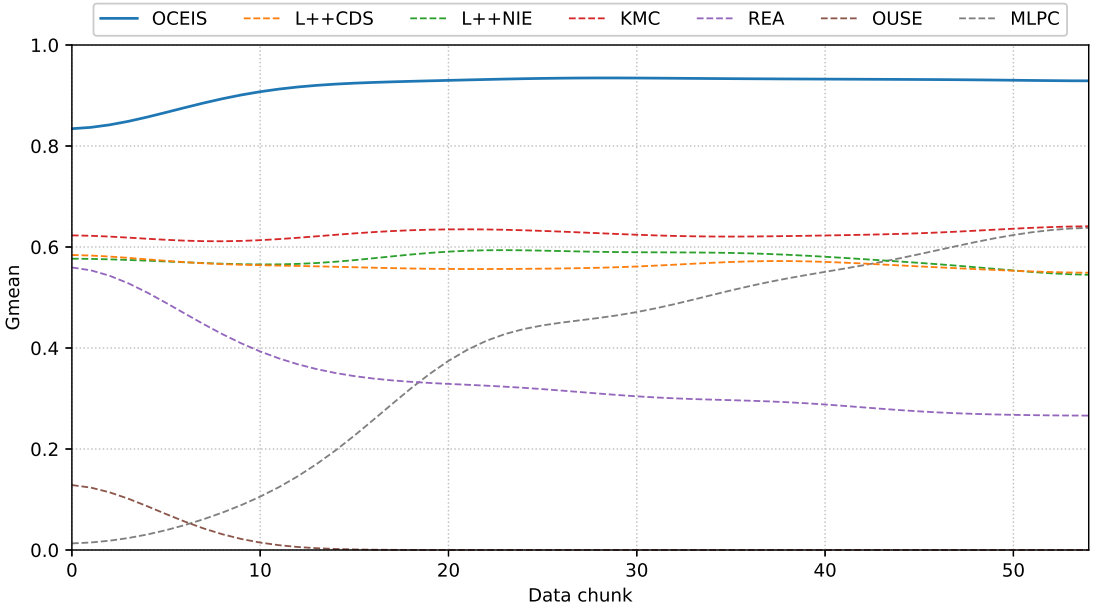


Figure 3.7: *$Gmean_s$ score over the data chunks for real stream shuttle-4-5vsA*

When analyzing the results, one should pay attention to the significant divergences in the performance of the proposed method for synthetic and real data streams. A large variety

characterized real data streams, while artificial streams were generated using one type of generator (of course, for different settings). However, the generated data streams are biased towards one type of data distribution, which probably was easy to analyze by some of the models, while the bias of the rest of them was not consistent with this type of data generator. Therefore, in the future, it should be carried out the experimental research on the expanded pool of synthetic streams generated by other different generators.

3.1.2 Lessons learned

To summarize the experimental evaluation conducted, the answers to the research questions posed earlier are presented below:

RQ1: Is it possible to design a one class ensemble method with a statistically better or equal predictive performance than the selected* state-of-the-art *methods for imbalanced data streams?

Based on the results obtained from reliable experiments, the formulated research question receives an affirmative answer. *OCEIS* achieves results at a similar level to the compared methods, but it is worth noticing that it performs best on the real data stream, which is its important advantage. Another advantage is that there is no tendency towards the excessive classification of objects into one of the classes. This was a problem in the experiments carried out using the REA and OUSE methods. Such "stability" contributes significantly to improving the quality of classification and obtaining satisfactory results.

3.2 One class support vector machine weighted ensemble

This subsection will describe the **O**ne **C**lass support vector machine **W**eighted **E**nsemble (*OCWE*) method for the imbalanced data stream classification. It is the chunk-based ensemble that employs *OCSVM* classifiers for binary classification problems. Ensemble diversity is ensured by feature space clustering, and a specially designed decision rule with weighted voting adds more versatility to this idea. The pictorial diagram 3.8 presents the main idea. Let us shortly explain its main components.

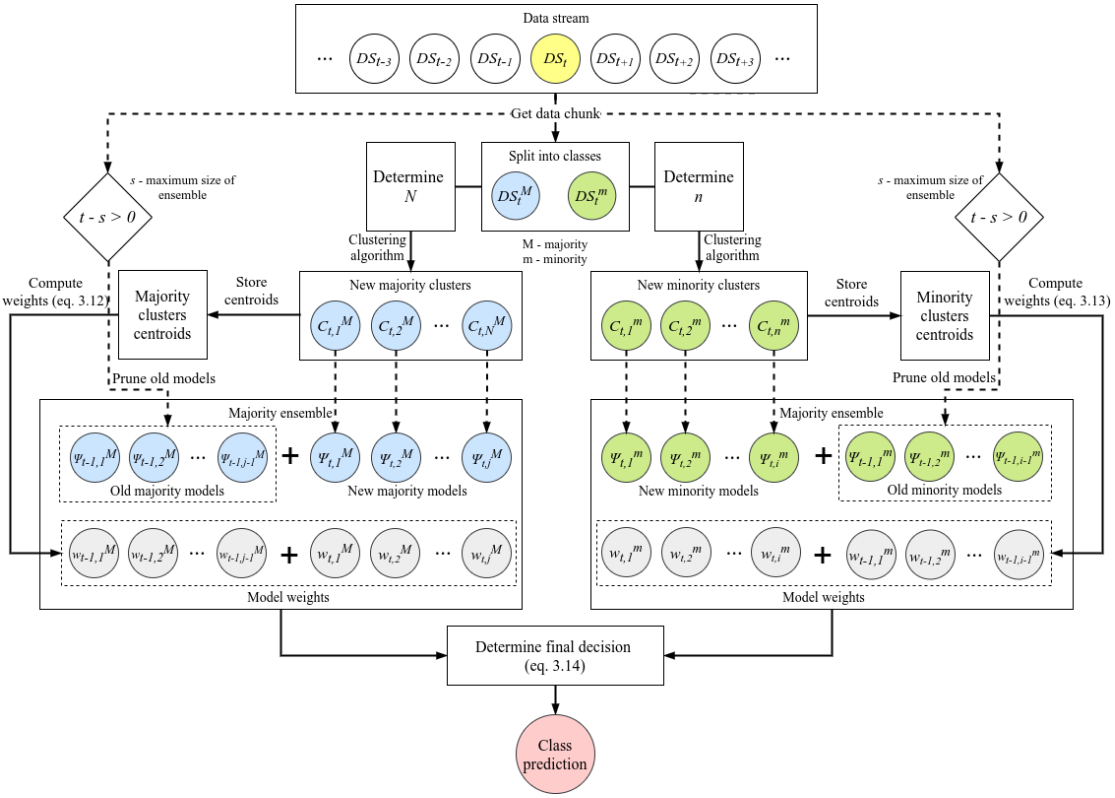


Figure 3.8: *Overview diagram of* OCWE *method*

OCWE is an extension of the **O**ne **C**lass support vector machine classifier **E**nsemble for **I**mbalanced data **S**tream (*OCEIS*) [132] (Sec. 3.1). Both methods are based on the ensemble of *OCSVM* and designed to classify imbalanced data streams However, *OCWE* introduces a few important changes:

- The main one is the weighted combination rule, where these weights are based on four different factors – the distance between models, deterioration, prediction metric, imbalance ratio of class. These factors have additional parameters to determine how much they influence the final decision.
- Clustering method and metric to determine the optimal number of clusters were chosen experimentally.

– The procedure for selecting the clusters for the majority examples was changed.

The whole method will be described step by step to explain better and justify these changes.

Ensemble

A crucial aspect of designing the ensemble method is to maintain a strong diversification of models. One of the well-known strategies is bagging that builds the ensemble using randomly selected samples from data. Sun et al. [239] proposed *Clustering Based Bagging Algorithm*, which first divides the samples using the K-means algorithm into clusters and then draws the samples from these clusters. *OCWE* provides diversification through clustered feature space. It means that each of the *OCSVM* models are trained on a different data cluster using the selected clustering algorithm. This approach was inspired by [144] where the authors showed that it improves the predictive quality of *OCSVM* ensemble. This idea significantly reduces the overlapping among the classifiers' decision regions in the ensemble. The procedure of the proposed method is described in Alg. 2.

Clustering

Each data chunk (DS_t) is divided into a minority set (DS_t^m) and a majority set (DS_t^M). Then each of these subsets is divided into clusters using a clustering algorithm (CA). *OCWE* was designed in such a way that the clustering method is a parameter. Unfortunately, this creates an issue – how to find the right number of clusters. One approach may be to determine this number arbitrarily. However, it is problematic in non-stationary streams, where the data distribution is affected by some changes. The initially determined number of clusters is unlikely to work well after the concept drift. Another solution is to use a metric that could how well such clusters were created. Having a metric, it is possible to determine (within a certain range) what number of clusters gets the best value and make a decision based on that for every chunk of data.

Therefore, an dynamic way of selecting the best number of clusters is introduced. Minority set is split multiple times ($C_{t,k}^m$) into a different number of clusters from 1 to K. Then the value of the metric (CM) is calculated for every solution. The choice of the final number of clusters (n) for minority class data is made based on the best value of the metric (CM). The metric is also a parameter that can use any function to calculate the value of consistency for the created clusters. A slightly different approach is used for majority class objects. Most often, the majority instances form a large subset of the data chunk. Searching for the best number of clusters using the metric may be

Algorithm 2 OCWE

Input:

DS – data stream
DS_t – t-th data chunk of data stream DS
s – maximum size of classifier ensemble
K – maximum number of clusters
CM – selected clustering metric
CA – selected clustering algorithm

Symbols:

DS_t^m – minority data set of chunk DS_t
DS_t^M – majority data set of chunk DS_t
N – number of majority clusters
n – number of minority clusters
$C_{t,k}^m$ – clusters of minority data DS_t^m
$C_{t,k}^M$ – clusters of majority data DS_t^M
$\Psi_{t,i}^m$ – model trained on $C_{t,i}^m$ cluster data using Ψ
$\Psi_{t,j}^M$ – model trained on $C_{t,j}^M$ cluster data using Ψ

Output:

Π^t – final ensemble for t-th data chunk

for $t = 1, 2, \ldots$ **do**
  Split DS_t into minority (DS_t^m) and majority (DS_t^M) data
  for $k = 1, 2, \ldots, K$ **do**
    $C_{t,k}^m \leftarrow$ Create k clusters with CA on DS_t^m
  end for
  $n \leftarrow \underset{k=1,2,\ldots,K}{\operatorname{argmax}} CM(\mathrm{C}_{t,k}^m)$
  for $i = 1, 2, \ldots, n$ **do**
    $\Psi_{t,i}^m \leftarrow$ Train $OCSVM$ model on $C_{t,i}^m$ cluster data
    $w_{t,i}^m \leftarrow$ Compute weight for $\Psi_{t,i}^m$ according to the Eq. 3.12
    $\Pi^m \leftarrow \Pi^m \cup \Psi_{t,i}^m$
  end for
  $N \leftarrow \left\lceil \frac{\|DS_t^M\|}{\|DS_t^m\|} \right\rceil$
  $C_t^M \leftarrow$ Create N clusters with CA on DS_t^M
  for $j = 1, 2, \ldots, N$ **do**
    $\Psi_{t,j}^M \leftarrow$ Train $OCSVM$ model on $C_{t,j}^M$ cluster data
    $w_{t,j}^M \leftarrow$ Compute weight for $\Psi_{t,j}^M$ according to the Eq. 3.13
    $\Pi^M \leftarrow \Pi^M \cup \Psi_{t,j}^M$
  end for
  if $t > s$ **then**
    $\Pi^M \leftarrow \Pi^M \setminus \Psi_{t-s,j}^M$
    $\Pi^m \leftarrow \Pi^m \setminus \Psi_{t-s,i}^m$
  end if
end for

computationally intensive. Preliminary experiments have shown that using an approach with a metric (CM) leads to the formation of many clusters. Unfortunately, in this case, numerous clusters are associated with numerous majority class models in the ensemble. As a result, the final decision of the committee is too biased towards the majority class. This trend was visible in the experiment results of *OCEIS* method [132]. The number of clusters is determined according to the formula below:

$$N = \left\lceil \frac{\|DS_t^M\|}{\|DS_t^m\|} \right\rceil \tag{3.3}$$

where N is a number of clusters for majority class data, $\|DS_t^M\|$ is a number of samples in the majority data chunk, and $\|DS_t^m\|$ is the number of samples in the minority data chunk, i.e., the number of majority clusters is equal to the Imbalance Ratio rounded up. Then clusters ($C_{t,k}^M$) are created.

Pruning

The clusters ($C_{t,k}^m$, $C_{t,k}^M$) are used to train new models ($\Psi_{t,i}^m$, $\Psi_{t,j}^M$) using $OCSVM$ which are forming the minority (Π^m) and majority (Π^M) ensemble. In each step, a few new models are added to the ensemble. It is vital to implement an efficient mechanism for pruning models. The main rule is to remove the oldest models from the ensemble when they exceed a predetermined threshold. When the ensemble size reaches a preset maximum threshold (s), the oldest models are removed. Such an approach avoids memory overflow and automatically adapts to the potential concept drift in a non-stationary data stream. This design can also process data streams that are theoretically infinite.

Weights

After pruning, the weights for each classifier are calculated based on four factors. The first one is the distance to the nearest classifier from the opposite class. The higher is the distance, the higher is the "competence" of this classifier. However, when two classifiers from different classes are very close to each other, their "competence" becomes much more smaller. Each model in the ensemble is trained on a data cluster ($C_{t,k}^m$, $C_{t,k}^M$) created on a data chunk. The relative distance between the models is estimated based on the centroid points ($CP_{t,k}^m$, $CP_{t,k}^M$) of these clusters. For the minority class model $\Psi_{t,i}^m$, this "competence" level is defined by a minimum Euclidean distance between $CP_{t,i}^m$ point to $CP_{t,k}^M$ points:

$$CL_{t,i}^{m} = \min_{k \in 1,\ldots,N} |CP_{t,i}^{m} - CP_{t,k}^{M}| \tag{3.4}$$

where t is the current data chunk number, i is the index number of minority model centroid point, k is the index number of majority model centroid point from 1 to N. Similarly, for a majority class models, "competence" level is defined by the minimum Euclidean distance between $CP_{t,j}^{M}$ point to $CP_{t,k}^{m}$ points:

$$CL_{t,j}^{M} = \min_{k \in 1,\ldots,n} |CP_{t,j}^{M} - CP_{t,k}^{m}| \tag{3.5}$$

where t is current data chunk number, j is the index number of majority model centroid point, k is the index number of minority model centroid point from 1 to n. Another factor is the quality of the classification, which is evaluated on the current data chunk. For minority class classifiers, it is *Recall* score:

$$S_{t,i}^{m} = \frac{tp}{tp + fn} \tag{3.6}$$

where tp is true positive and fn is false negative. For majority class classifiers $Specificity$ score:

$$S_{t,j}^{M} = \frac{tn}{tn + fp} \tag{3.7}$$

where tn is true negative and fp is false positive. The next factor is the age ($A_{t,i}^{m}$, $A_{t,j}^{M}$) of each classifier. Here the rule is quite simple. For each classifier, the age is counted. The factor A is inversely proportional to the difference of age of the model and the actual chunk number. It means that the oldest classifiers have the smallest weight. For minority class models, this value is computed as:

$$A_{t,i}^{m} = \frac{1}{T^{i} - t} \tag{3.8}$$

where T^{i} is the index number of the data chunk on which the model ($\Psi_{t,i}^{m}$) was created and t is the current chunk index number. For majority class models:

$$A_{t,j}^{M} = \frac{1}{T^{j} - t} \tag{3.9}$$

where T^{j} is the index number of the data chunk on which the model ($\Psi_{t,j}^{M}$) was created and t is the current chunk index number. The last factor is the imbalance ratio in a given data chunk. The higher the imbalance ratio, the higher is the weight of the minority models, and simultaneously the majority models weight decreases. Minority models usually have smaller decision regions and are less numerous, which leads to a strong bias of the whole ensemble decision toward the majority. For minority class models, the imbalance factor is the imbalance ratio:

$$IF_{t}^{m} = \frac{\left\|DS_{t}^{M}\right\|}{\left\|DS_{t}^{m}\right\|} \tag{3.10}$$

where t is the current data chunk number. For majority class models, it is defined as the inverse value of the imbalance ratio:

$$IF_{t}^{M} = \frac{\left\|DS_{t}^{m}\right\|}{\left\|DS_{t}^{M}\right\|} \tag{3.11}$$

To calculate the final weights for minority models, the following weighted sum is used:

$$w_{t,i}^{m} = \alpha \times CL_{t,i}^{m} + \beta \times S_{t,i}^{m} + \gamma \times A_{t,i}^{m} + \delta \times IF_{t}^{m} \tag{3.12}$$

Similarly, for the majority models:

$$w_{t,j}^{M} = \alpha \times CL_{t,j}^{M} + \beta \times S_{t,j}^{M} + \gamma \times A_{t,j}^{M} + \delta \times IF_{t}^{M} \tag{3.13}$$

where $(\alpha, \beta, \gamma, \delta)$ are responsible for the proper weight tuning.

Prediction

Designing a suitable combination rule is an essential element of each ensemble method. It is important for better performance and satisfactory predictive quality. Methods such as *OCEIS* or *OCWE* that use one-class classifiers for binary problems require special attention. Typical solutions based on majority voting or soft voting cannot produce a good predictive ability. Both methods use the original combination rule based on the classifiers' decision boundary distance to predict samples. The main change that has been introduced in the *OCWE* method is the decision weighting of individual classifiers

in the ensemble. Many ensemble methods use weights to change the influence of decisions from individual models [36, 258].

Let $f_{t,i}^m$ denotes decision function of classifier $\Psi_{t,i}^m$ (Eq. 2.10) and $f_{t,j}^M$ of classifier $\Psi_{t,j}^M$ respectively. The final decision of the proposed classifier Ψ is made according to the following classification rule:

$$\Psi(x) = \begin{cases} \text{minority class} & if \max\limits_{i \in \Pi^m} \left(w_{t,i}^m f_{t,i}^m(x) \right) \geq \max\limits_{k \in \Pi^M} \left(w_{t,k}^M f_{t,k}^M(x) \right) \\ \text{majority class} & \text{otherwise} \end{cases} \tag{3.14}$$

Computational complexity analysis

The computational complexity of the *OCWE* method consists of several factors that are specific parts of the whole algorithm. In order to determine the complexity of the whole method, it is necessary to determine the complexity for these individual elements and then make a linear combination of them. The data clustering method has a relatively significant impact to final complexity. Because this is a certain parameter for the *OCEIS* algorithm, it must be assumed that the K-means method will be used. It has a complexity equal to $O(kt_i n_s)$ [123] where n_s is the number of objects, k is the number of clusters, and t_i is the number of iterations. Another important factor is the procedure that determines the weights for each model. The most aggravating part is determining the distance between centroids using the Euclidean. Under the worst-case assumption, it can be determined that this complexity equals $O(c_p)$ [8], where c_p is the number of centroid pairs that will be compared to each other. The last piece that affects the complexity of the whole algorithm is the base classifier, which in the case of this method is OCSVM. Its complexity is equal to $O(n_s^3)$ [275], where n_s is the number of objects. Concluding, the complexity of the proposed method depends on several factors. The linear combination of these fragments allows to determine the final computational complexity equal to $O(n_s^3 + kt_i n_s + c_p)$. The complexity for the *Learn++CDS* method is $O(n_s^3 + n_s \log_2 n_s)$ [62] [73] [276]. This means that the proposed *OCWE* method has lower theoretical computational complexity than *Learn++CDS*.

3.2.1 Experimental evaluation

This subsection will describe the experiments' setup and results to explore the properties of the proposed method and the prediction quality for imbalanced data streams compared

to selected *state-of-the-art* methods. Let us formulate the following research hypotheses:

RQ1: What is the best setup of clustering methods and cluster consistency metric in the *OCWE* method?

RQ2: What is the best setup of weight parameters in the *OCWE* method?

RQ3: How flexible is *OCWE* to the dynamic imbalance ratio and concept drift?

RQ4: What is the predictive quality of *OCWE* in comparison to the *state-of-the-art* methods?

Setup

All experiments and the proposed method have been implemented in the Python programming language. The implementation and the full set of results are publicly available on the Github repository[2]. The research was conducted on imbalanced data streams from two different generators – stream-learn [150] and *MOA* [24]. Tab. 3.3 presents parameters such as a number of objects, chunk size, imbalance ratio, noise level or drift type of generated synthetic data streams.

Table 3.3: *Parameter setup for data stream generators*

Parameter	Value				
Number of samples	100000				
Number of chunks	200				
Chunk size	500				
Number of classes	2				
Number of features	10 (8 informative + 2 redundant)				
Number of drifts	1		5		
Concept drift types	sudden		incremental		
Random state	1111	2222	3333	4444	5555
Stationary imbalance	5%	10%	15%	20%	30%
Dynamically imbalance	5%	10%	15%	20%	30%

(a) *stream-learn*

Parameter	Value				
Generator type	RandomRBFGenerator				
Number of samples	100000				
Number of chunks	200				
Chunk size	500				
Number of classes	2				
Number of features	10 (8 informative + 2 redundant)				
Number of drifts	1		5		
Concept drift types	sudden		incremental		
Random state	1111	2222	3333	4444	5555
Stationary imbalance	5%	10%	15%	20%	30%

(b) *MOA*

Additionally, selected imbalanced streams of real origin were used (Tab. 3.4). Data chunks were processed in a *test-then-train* manner [24]. This means that every data chunk (except the first one) was used initially for evaluation and then for training. One chunk contains 500 objects. All tested methods were set to maximally store data and

[2]Repository link: https://github.com/w4k2/ocwe

models from the last 10 data chunks. The *OCWE* method has been compared with the selected *state-of-the-art* chunk-based ensemble methods:

- *REA* – Recursive ensemble approach [51]
  - Number of estimators = 10
  - Post balance ratio = 0.5
- *KMC* – K-mean clustering undersampling ensemble [264]
  - Number of estimators = 10
- *L++CDS* – Learn++CDS [62]
  - Number of estimators = 10
  - Parameter a = 0.5
  - Parameter b = 10
- *L++NIE* – Learn++NIE [62]
  - Number of estimators = 10
  - Parameter a = 0.5
  - Parameter b = 10
- *OUSE* – Over and undersampling ensemble [89]
  - Number of estimators = 10
  - Fixed sampling = True
- *MLPC* – Multi-layer perceptron classifier
  - Hidden layers = 1
  - Neurons in hidden layer = 10

Table 3.4: *Real datasets used in experimental evaluation*

Dataset	SAMPLES	FEATURES	IR
covtypeNorm-1-2vsAll	266 000	54	3.91
poker-lsn-1-2vsAll	360 000	10	9.69

The experiments were evaluated on the basis of five different metrics – $Gmean_s$ [152], $F_1 score$ [235], $Recall$, $Specificity$, $Precision$. Experiments were performed with four base classifiers using implementations from scikit-learn [200] machine learning library. List of selected base classifiers and their acronyms:

- *k-Nearest Neighbors* (*KNN*)
- *Support Vector Machine* (*SVM*)
- *Gaussian Naive Bayes* (*GNB*)
- *Decision Tree CART* (*DTC*)

Experiment 1 - Clustering method and metric

The first experiment focuses on testing different clustering algorithms (CA) and metrics for estimating potentially the best number of clusters (CM). These experiments were conducted on streams from two generators: *MOA* and stream-learn. Tab. 3.5 shows selected parameters to generate synthetic streams, which were used in this experiment. The obtained results are presented below in the form of radar charts. Both graphs show the average classification quality expressed in five different metrics – F_1score, $Gmean_s$, $Precision$, $Recall$, $Specificity$. More detailed results in the form of waveform charts can be found on the repository.

Parameter	**Value**		
Generator	MOA		stream-learn
Number of samples	100000		
Number of chunks	200		
Chunk size	500		
Number of classes	2		
Number of features	10 (8 informative + 2 redundant)		
Number of drifts	1		
Concept drift types	sudden		incremental
Imbalance ratio	10%	20%	30%
Label noise	0%	1%	5%

Table 3.5: *Parameter setup for data stream generators*

Fig. 3.9a shows the results for the tested five different data clustering methods – Agglomerative Clustering (*Aglom*), Spectral Clustering (*Spect*), Mini Batch K-means (*MBKM*), *KMeans*, *Birch*. The main criterion for selecting the above methods was determining the desired number of clusters and a certain variety of segmentation techniques. The exception to this diversity was K-means and Mini Batch K-means methods because they are very similar. They differ more in implementation than conceptual. Spectral Clustering uses information from the eigenvalues of special matrices built from the data set. Agglomerative Clustering is a hierarchical clustering method using a bottom-up approach, which starts from clusters with only one sample and then merges them in. Birch constructs the tree's data structure, in which cluster centers are read out of the leaf.

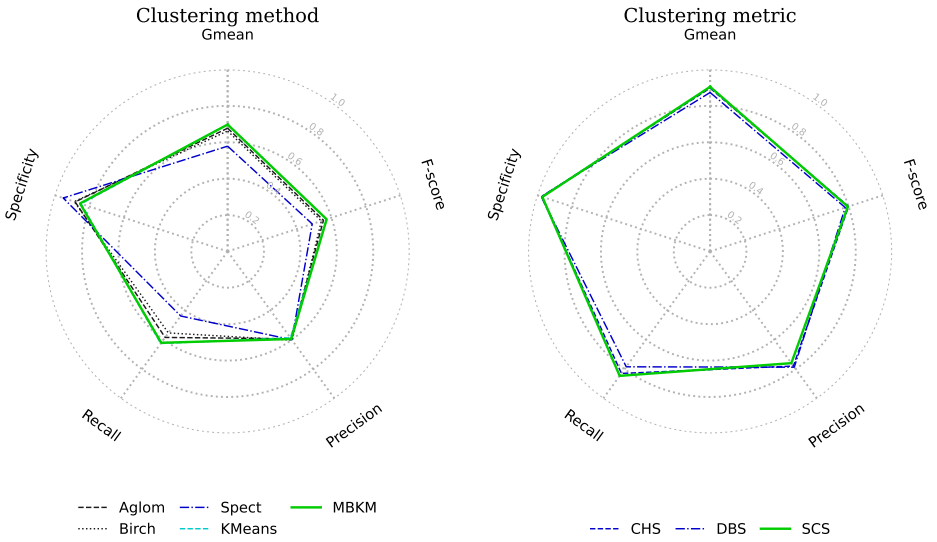


Figure 3.9: *Clustering method and metric setup on radar plots*

The obtained results (Fig. 3.9a) indicate that all tested methods for data clustering obtain quite similar metrics. The Spectral Clustering method, which reaches the highest value of $Specificity$, stands out slightly. Unfortunately, it is reflected in a considerable loss in the *Recall*. I chose Mini Batch K-means as the best method because it has the best values for all metrics except $Specificity$.

Fig. 3.9b shows the results for the three selected cluster consistency measures. First, it is a Silhouette Coefficient score (SCS) [165] based on Silhouette Value (SV) [217]. This metric describes how similar objects are to their cluster compared to other clusters. Another is the Calinski and Harabash score (CHS) [43], also known as the Variance Ratio Criterion. It is an average ratio between the within-cluster dispersion and the between-cluster dispersion. The last metric is Davies-Bouldin score (DBS) [56], which is an average ratio of within-cluster distance to between-cluster distance. The results (Fig. 3.9b) are even closer than in the previous experiment. However, after a closer look it can be seen that the SCS metric stands out very gently for the best results for all metrics except *Precision*. Therefore, this metric was chosen for further experiments.

Experiment 2 - Weight parameters setup

The next experiment is selecting the best values for the parameters $\alpha, \beta, \gamma, \delta$, which define how much weight factors influence the final decision. Only four representative data streams were selected for this experiment. The reason for this limitation is the high computational complexity of the parameter search process. A stream-learn generator

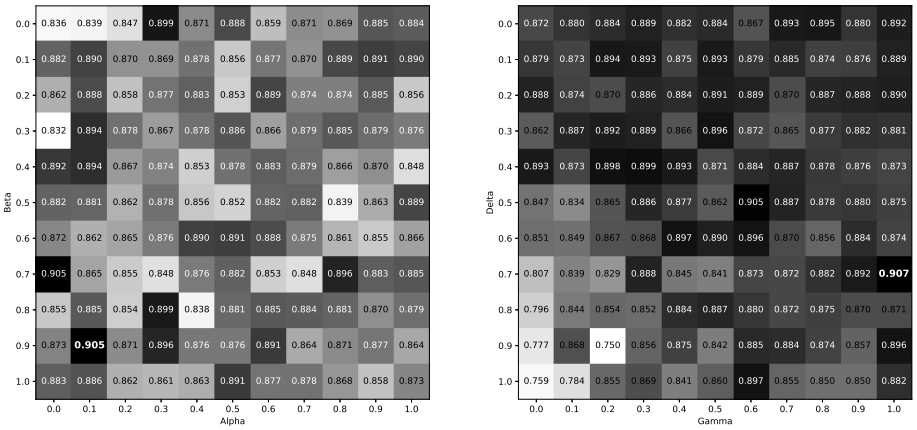


Figure 3.10: *Weight parameters setup for $Gmean_s$ on the stream-learn data stream*

generated the first stream. The generated stream has a hundred thousand objects, one sudden concept drift, 1% label noise, and 15% imbalance ratio. The second stream has identical settings, although it is generated with the *MOA* generator. The next two are the "covtypeNorm-1-2vsAll-pruned" and "poker-lsn-1-2vsAll-pruned" real streams described in Tab. 3.4.

Parameters were combined in pairs – α, β and γ, δ. Then, a grid search was carried out for values from 0 to 1 with a division into 0.1 steps for these pairs. Several chunks were taken from each stream to train the classifier, and then the next chunks were used to evaluate the results for the selected settings. For streams generated synthetically, 20 chunks of 200 objects were used for training and 5 chunks of 200 objects each for testing. Real streams have a small number of minority class objects; thus 5 chunks of 2000 objects each for training and 2 chunks of 2000 objects for testing were used. In further experiments, the samples used here were excluded.

The obtained results are averaged and plotted in the grid tables (Fig. 3.10a and Fig. 3.10b). Based on these results obtained for each data stream, the best parameter values were selected and, using such settings, and subsequent experiments were executed.

Experiment 3 - Dynamic imbalance

The following experiment is intended to study how the proposed method works on streams with a dynamic class imbalance ratio. For comparison, selected *state-of-the-art* methods were used. The data for this experiment was generated using a stream-learn generator (Tab. 3.3). In this part is forced to limit experiments to only one generator because *MOA* cannot generate dynamically imbalanced data.

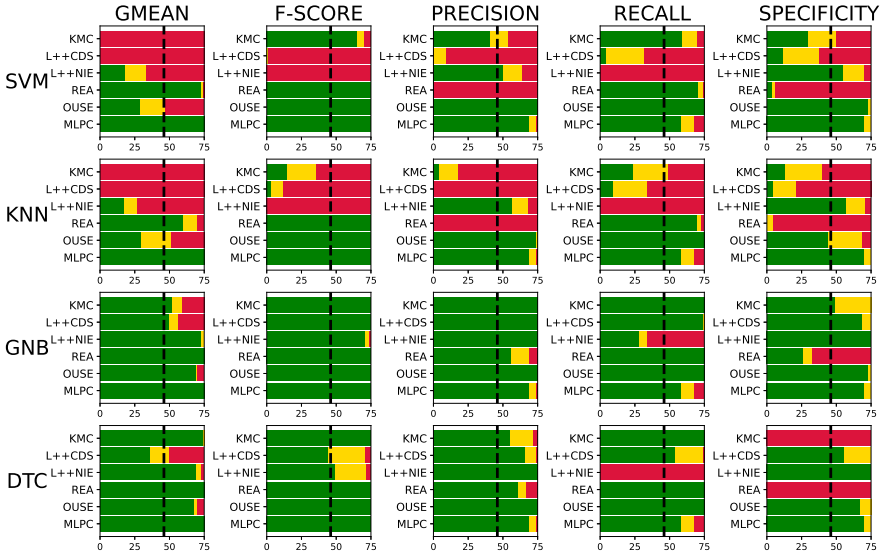


Figure 3.11: *Wilcoxon pair rank-sum tests for synthetic data streams with incremental concept drift and dynamic imbalance ratio. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*

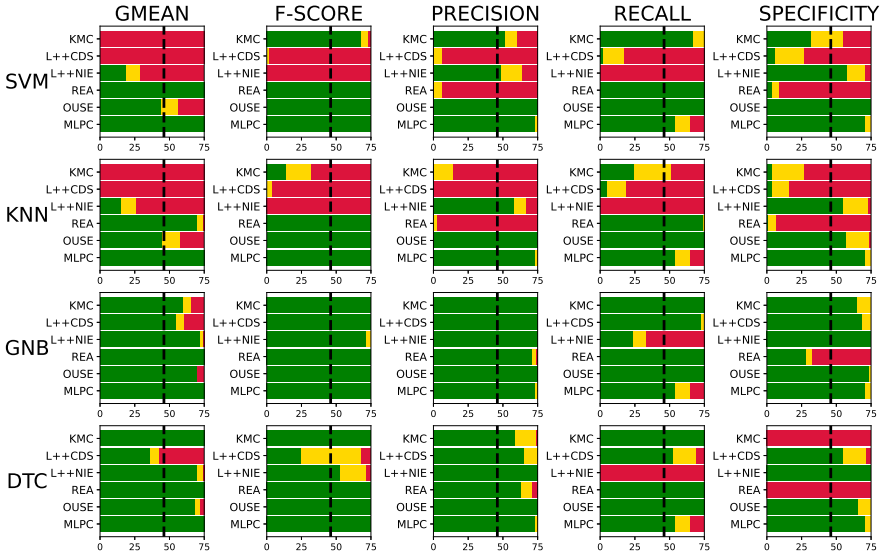
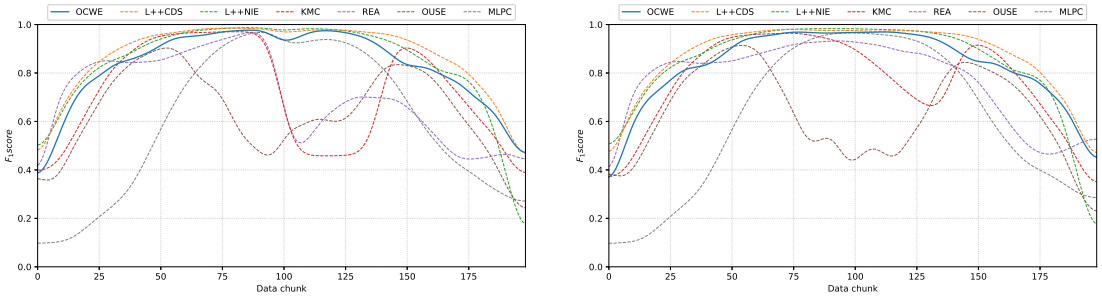


Figure 3.12: *Wilcoxon pair rank-sum tests for synthetic data streams with sudden concept drift and dynamic imbalance ratio. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*

This experiment's main aim is to show how the proposed method can deal with the dynamic imbalance ratio compared to other selected methods. A large number of used data streams forces to make some aggregation of results. The results will be presented in the form of metrics analysis using the Wilcoxon rank-sum pair statistical tests. The Fig. 3.11 shows such study for incremental drift streams and the Fig. 3.12 for sudden drift streams. The *OCWE* is not the worst in comparison with the tested methods. Throughout all the metrics, the *OCWE* performs the best with the *GNB* and *DTC*



(a) Sudden drift *(b) Incremental drift*

Figure 3.13: *$F_1 score$ metric over the data chunks for synthetic data stream with 1% label noise and dynamic imbalance ratio (from 5% to 95%) using the* SVM *base classifier*

base classifiers. Looking closer at the results can be seen that for *SVM* and *KNN* the Learn++CDS method performs better. The proposed method has a slight advantage in *Precision* and *Specificity*, indicating a better ability to classify majority class objects.

Fig. 3.13a and Fig. 3.13b shows the quality of the $F_1 score$ for each data chunk on selected synthetic streams with dynamic class imbalance. The label noise is at 1% level, and the imbalance ratio changes from 5% to 95%. For sudden and incremental drift *OCWE* obtains quite similar results to other methods. Sometimes it is gaining a little advantage. In a stream with a sudden drift, the *REA* and *KMC* methods at the drift spot significantly lose quality. They rebuild these losses at the end of the data stream. The *OUSE* method loses quality around the 50th data chunk due to a change in imbalance ratio and not to drift because it appears in the middle of the entire data stream near the 100th data chunk. For data streams with incremental drift, some quality drops are also less noticeable by most tested methods. More plots showing similar waveforms for the rest dynamic imbalanced streams and metrics can be found in the project repository.

Experiment 4 - Static imbalance

In this experiment, the methods are tested on the static imbalance ratio streams with the concept drift. Most of the generator settings are similar to the previous experiment (Tab. 3.3). The important change is using two independent data generators – stream-learn and *MOA*.

As in the previous experiment, such a stream amount generates a substantial number of results. The results will be presented in the form of metrics analysis using the Wilcoxon rank-sum pair statistical tests. Fig. 3.14 shows these results for incremental drift streams and Fig. 3.15 for the sudden drift. It can be observed that the *OCWE* method does not

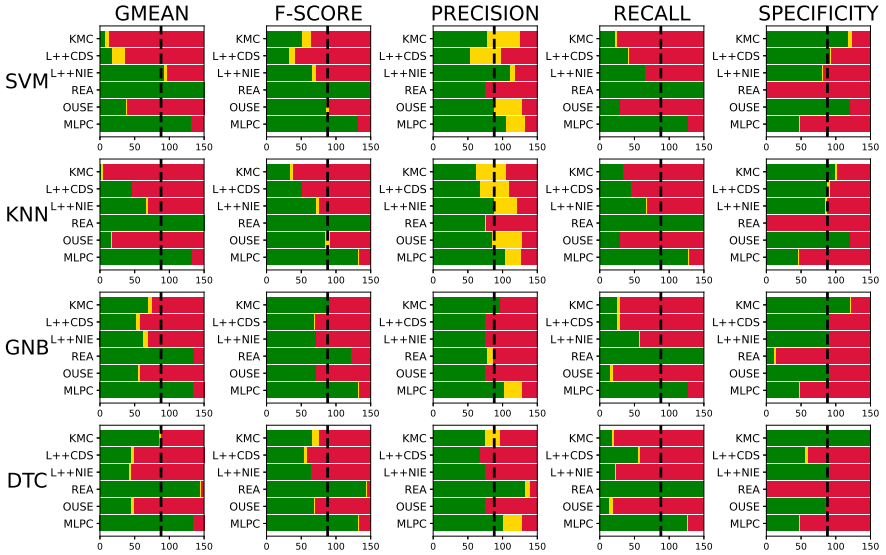


Figure 3.14: *Wilcoxon pair rank-sum tests for synthetic data streams with incremental concept drift. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*

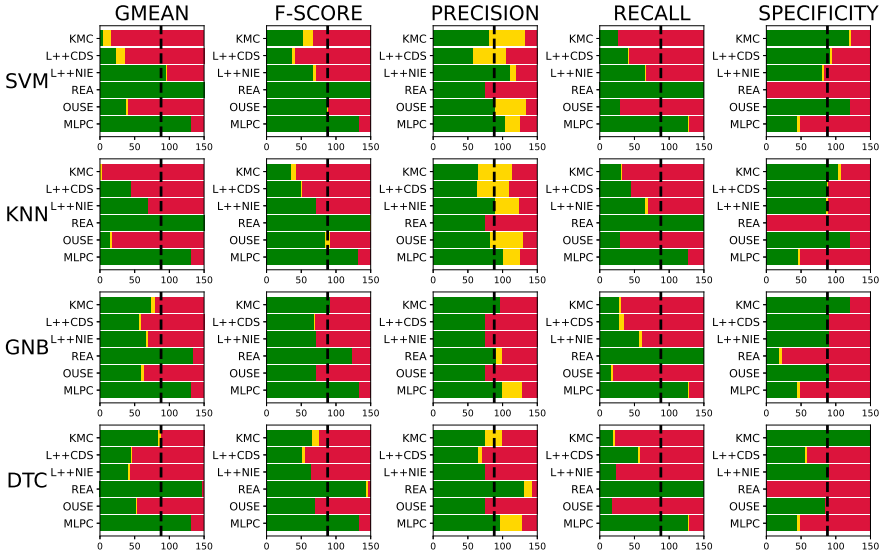
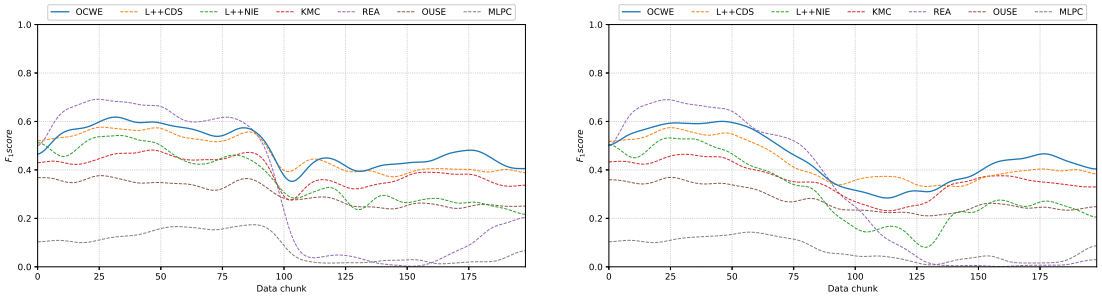


Figure 3.15: *Wilcoxon pair rank-sum tests for synthetic data streams with sudden concept drift. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*

work on the streams with static imbalance so well as previously. The best results among all metrics are obtained for *Precision*. The situation is not much worse for F_1score. There are also some ties for chosen metrics, which means that the method works at a similar level to others. In most of the presented results, the *OCWE* method usually achieves better predictive quality than the *REA* method and the *MLPC* method.

Fig. 3.16a and Fig. 3.16b shows the quality of the F_1score for each chunk from two selected synthetic streams with the 5% imbalance ratio and 1% label noise. One can see



(a) Sudden drift *(b) Incremental drift*

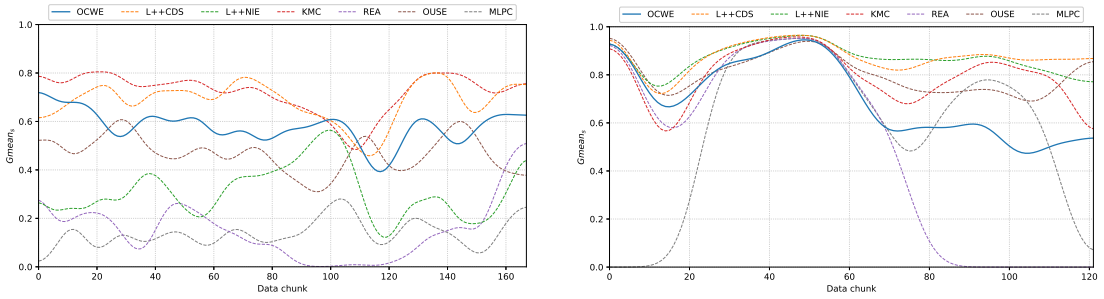
Figure 3.16: *F_1score over the data chunks for synthetic data stream with 1% label noise and static imbalance ratio (5%) using the* SVM *base classifier*

that the *OCWE* method did not achieve the worst results, and sometimes it is even the best method among other tested. For a stream with a sudden drift, the *REA* method is initially the best. Near the 100th data chunk, where concept drift appears, the quality decreases drastically and is not rebuilt. *OCWE* and Learn++CDS achieve similar good results. It is worth noting that in the most of the data stream, the *OCWE* perform better than Learn++CDS. It refers to the results before and after the occurrence of concept drift.

A similar situation could be observed for the incremental drifted streams where the *REA* method is initially the best. When the concept drift changes are more intense, it loses significantly on predictive quality. Learn++CDS and *OCWE* perform the best overall. Before the drift, *OCWE* gets better results, but the Learn++CDS method is better after the drift. More plots showing similar waveforms for the rest imbalanced data streams and metrics could be found in the project repository.

Experiment 5 - Real data

Previous studies have focused mainly on the testing the method under various and precisely defined conditions. Selection of various settings, such as concept drift types, label noise, imbalance ratio, etc., provides the opportunity to test the method in a vast range of data stream characteristics. It is equally important to carry out experiments with data from real life. Unfortunately, real data streams with a significant high imbalance ratio are rare. Some methods allow artificially changing the imbalance ratio or modify static data sets as streams with artificial concept drift. However, such modifications cause large disturbances. It is most reflected in changes to the original data distribution, which results in semi-synthetic streams.



(a) Stream poker-lsn-1-2vsAll *(b) Stream covtypeNorm-1-2vsAll*

Figure 3.17: *$Gmean_s$ over the data chunks for real data stream using the* SVM *base classifier*

For this reason, this work will focus only on two selected real data streams [44] (Tab.3.4). Nevertheless, these data streams require two small modifications. First, it is necessary to binarize the classes by combining two selected groups of classes. The next change is to select a particular part from the entire stream to avoid the complete vanishing of any class because selected *state-of-the-art* reference methods use binary class-based classifier models. With the prepared data, it is possible to experiment with how the proposed method works on real imbalanced data streams.

As mentioned before, the selection of hyperparameters for the *OCWE* method was made before the selected experiments. It was enough for the synthetic streams to generate an additional data stream on which this process occurred. With real streams, this option is not possible. This problem is solved so that the first ten thousand objects were used only for setting parameters. However, the final experiment to check the performance was done without this beginning subset of data. The results are presented as waveforms (Fig. 3.17). The scores for the $Gmean_s$ will be discussed in more detail. The rest plots for other metrics are available on the repository.

OCWE for both real data streams is not the worst than the other tested methods. At first sight, the proposed method for the "poker-lsn-1-2vsAll" stream gets the third-best result. Around 115th data chunk show some decay, most likely caused by concept drift. A drop in quality is noticed but is restored in subsequent iterations. Apart from this incident, the stream *OCWE* method keeps the result stable without significant drops. Learn++CDS and *KMC* were the best methods, with the quality being at a similar level.

Slightly worse results were achieved by the *OCWE* method for "covtypeNorm-1-2vsAll" stream. In the initial phase of the stream, the results are very similar to the other methods. At the 60th data chunk is most likely a concept drift, which causes a significant drop in scores. Unfortunately, the proposed method does not return to performance before the drift. The Learn+++NIE and Learn++CDS methods achieved the best results.

3.2.2 Lessons learned

One Class Weighted Ensemble is a method that extends the idea of using the one-class classifier ensemble to classify binary imbalanced data streams. Diversification of the ensemble is ensured by decomposition into one-class subsets and clustering of this data. The changes introduced compared to the original proposal (*OCEIS*) [132], indicate a positive impact on the classification quality. To summarize the experimental evaluation conducted, the answers to the research questions posed earlier are presented below:

RQ1: What is the best setup of clustering methods and cluster consistency metric in the OCWE method?

The best clustering method and the best metrics for determining the created clusters' consistency were selected. Experiments with a small advantage suggest the Mini Batch K-means as the best method and the Silhouette Coefficient as the best metric. The rest of the experiments were performed using this method and the metric. It is impossible to find one optimal configuration of parameters that determine the strength of the component weights (α, β, γ, δ). These parameters were chosen for the selected data streams, taken as representative examples of specific streams. However, it is important to note the *OCWE* is a parameterized method and leaves the possibility of making modifications depending on the current problem under examination.

RQ2: What is the best setup of weights parameters in the OCWE method?

OCWE introduces the weights that determine the final decision of the classifiers' ensemble. The performed research shows that the parameterization of these weights and selecting appropriate settings improve the prediction ability. At the same time, it is worth noting that influence of each parameter depends on what kind of data is being processed. On the one hand, it does not allow to find one ideal setting for all problems and forces to optimize the parameters each time. However, this has a positive impact on the overall classification performance and the ability to adapt to various problems.

RQ3: How flexible is OCWE to the dynamic imbalance ratio and concept drift?

Experiments carried out on a relatively large set of generated streams allowed to thoroughly examine the proposed method's behavior. The waveform charts present the most information about the overall quality. The analysis of these graphs allows concluding

that the *OCWE* method is a relatively high-quality ensemble method for imbalanced data streams. It handles the data with the dynamic imbalance ratio very well. It is worth noting that the presented plots demonstrate quite a good ability to classify data streams, where the imbalance ratio is equal to 5%, and there is 1% label noise. Such results suggest that the method can handle difficult data – strong imbalance, noises, concept drift, and dynamic imbalance ratio. At the same time, *OCWE* can perform fairly well with imbalanced streams of real origin.

RQ4: What is the predictive quality of OCWE in a comparison to the* state-of-the-art *methods?

Wilcoxon pair rank-sum tests provided an extensive comparison of the *OCWE* with other selected *state-of-the-art* methods. By analyzing the obtained results, the method proposed in this article achieves results that are very similar to others. Overall the Learn++CDS seems to be the one of the best methods. However, under certain conditions, the predictive performance of the *OCWE* is better. The most significant advantage has been gained on dynamic imbalanced data streams. At the same time, *OCWE* has the highest number of Wilcoxon pair test wins for the *Precision*. It may indicate a slight skew towards the majority class. However, *OCWE* performs very well at classifying streams with a very high imbalance ratio.

Chapter 4

Hybrid data preprocessing methods

This chapter introduces a family of methods for classifying imbalanced data streams, which are relying on the exploitation of data preprocessing methodologies. These proposed approaches are mainly based on a data accumulation technique for semisynthetic data oversampling. A robust experimental analysis supported by statistical validation of the obtained results is also presented.

4.1 Deterministic sampling classifier

Classifying imbalanced data streams involves a few challenges. First, there must be adequate mechanisms to perform appropriate data balancing. This can be done by using cost-senstive methods or approaches based on data preprocessing. **D**eterministic **S**ampling **C**lassifier (*DSC*) is a method designed to classify binary imbalanced data streams that will focus on the latter solution. Furthermore, it is very important to provide an appropriate technique to the problem of non-stationary data streams, which are constantly changing and require model adaptation. A very common way is to use concept drift detection methods and then, once it is detected, rebuild the whole model. This approach works especially well when a data stream with sudden concept drifts is considered. Another approach is to use classifiers that have the ability to update their knowledge without losing information acquired in previous data chunks. However, this approach requires models that are able to extend their knowledge. Unfortunately, most known models do not have such a capability, which excludes them from the application.

DSC method focuses on using an appropriate data accumulation technique, so that it is possible to use this type of classifiers. The main idea is based on accumulating data from previous chunks and in this way, although the model is rebuilt completely from scratch, it retains knowledge from samples that appeared several data chunk earlier. At the same

time, this technique also solves the problem of imbalanced data. It collects patterns from a previous data chunk to perform semisynthetic oversampling of the data. Such a solution is an important part of the proposed method. First, it is worth noting that enhancing minority class objects with data from previous data chunks is a very interesting oversampling solution. The method does not remove data from the majority class which happens in the undersampling approach. Furthermore, completely new patterns are not created, which in a certain way try to mimic the current distribution of the data to best match the new samples to the current data chunk. This is a very special case of oversampling, which uses an existing distribution of data that forms a given data stream. It is important to note that unlike other methods that use similar solutions [52], the stored data comes from the minority class and the majority class (Fig. 4.1).

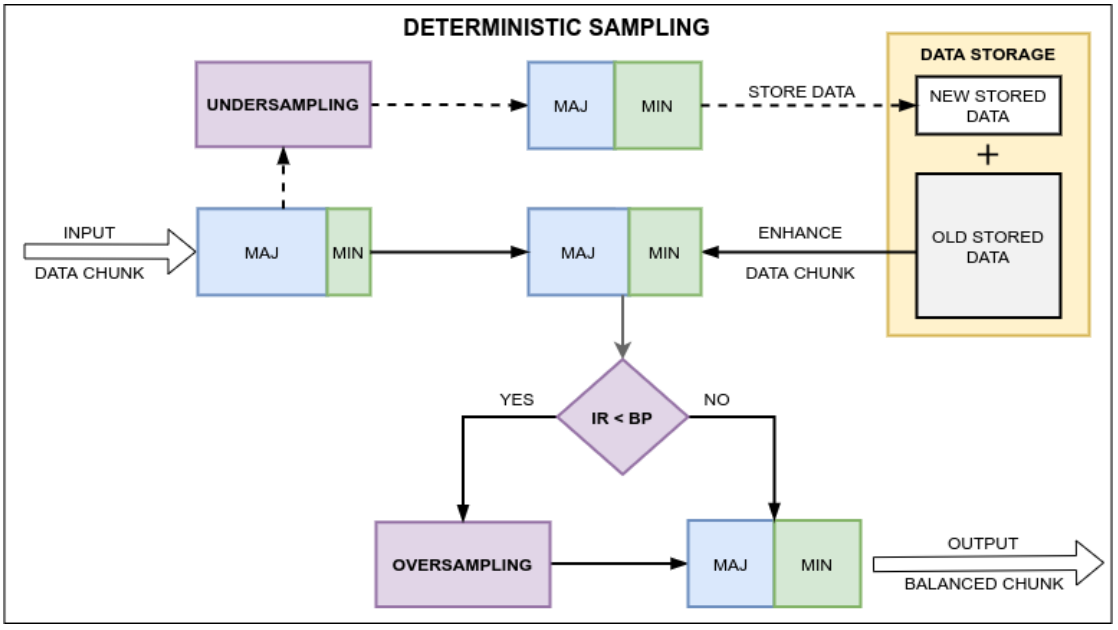


Figure 4.1: *Deterministic Sampling Classifier flow diagram*

Some proposed approaches attempt to do an matching of accumulated samples to the current data chunk through classification methods [51]. Other approaches focus on the distances of the data collected to the current data chunk [211]. However, a valuable approach that may allow such oversampling to have more potential is to include information about the majority class. At least a certain part that will preserve the relevant relationships between classes in the learned model. Otherwise, one may end up with a situation where some minority patterns used for amplification are patterns that significantly overlap the majority class and disrupt the decision boundary. Without this knowledge, a correct classification model will not be trained. Another important assumption of this approach is to process the data stream in a chunk-based manner. This means that objects arrive in certain blocks of data containing sets of patterns. This approach uses only one model, which is rebuilt from scratch on each new data chunk. Due to the

fact that previous data is stored, there is no need for models that have the ability to be updated.

It is worth noting that the presented *DSC* approach has a wide set of hyperparameters. Such a solution increases the ability of the method to adapt depending on the considered problem. According to Wolpert's theorem [268], there is no single method that will always works best under different conditions. However, by increasing the ability for flexible adaptation to the problem, much more better results can be achieved than with a rigidly planned solution focused strictly on one type of problem. This philosophy has also guided the design of this approach, where most of the used elements that compose the entire classifier allow for minor or major adjustments. One should be also aware that in order to use the full potential of the method, it is necessary to perform a hyperparameters calibration process. This step should be repeated each time the model is implemented in production.

Algorithm

The procedure of learning a new model is carried out in two phases (Alg. 3). First, undersampling of the data is performed. The method of undersampling is one of several parameters of this method, which can be freely chosen. Then, the data processed in this way, which have an equal distribution of classes, are stored in the Accumulated Data Storage (ADS). This is a certain reservoir that stores S of recent data chunks. This number is also a parameter that can be set arbitrarily. At this point, the first phase is complete.

In the second phase, the data chunk goes through a couple of processing steps so that it finally goes to the selected base classifier to create a new model. First, it is boosted with all collected data in ADS. It is obvious that in the first iteration, there is no collected data. Therefore, one essential part of this method is the additional fully synthetic oversampling. It is performed using the selected method for oversampling, which is also a changeable parameter. After more data chunks arrive ($t > S$) and samples are accumulated in ADS, a strong enough set is created that additional oversampling is no longer necessary. This is regulated by another parameter (BP) that determines how high the imbalance level of the data chunk after enhancing from ADS does not require oversampling of the data. The data chunk prepared in this way goes to the base classifier (Ψ) and the second phase comes to an end.

Algorithm 3 DSC

Input:

DS – data stream
DS_t – t-th data chunk of data stream DS
BCL – base classifier learning method
OSM – oversampling method
USM – undersampling method
BP – balance parameter
S – max size of stored data chunks

Symbols:

ADS – accumulated data storage
DSU_t – undersampled t-th data chunk

Output:

Ψ_t – final model for t-th data chunk

$ADS = \emptyset$
for $t = 1, 2, \ldots$ **do**
    $DSU_t \leftarrow$ Undersample DS_t using USM
    $DS_t \leftarrow DS_t \cup ADS$
    if IR of $DS_t > BP$ **then**
        $DS_t \leftarrow$ Oversample DS_t using OSM
    end if
    $\Psi_t \leftarrow$ Train model on DS_t using BCL
    $ADS \leftarrow ADS \cup DSU_t$
    if $|ADS| > S$ **then**
        $ADS \leftarrow ADS \setminus DSU_{t-S}$
    end if
end for

Computational complexity analysis

The computational complexity of the *DSC* consists of several components. The most important part of this method is ADS. There are only two operations performed on ADS – writing and reading. It takes $O(1)$ time to add a new item to the list of any size [54]. When saving new data, the time complexity can not exceed $O(n_s)$, where n_s is number of objects. Reading data from ADS and adding it to the current training data chunk does not exceed the complexity of $O(Sn_s)$, where S is the maximum number of chunks stored in the memory. Then the complexity is affected by the chosen method for undersampling and oversampling. Assuming that random undersampling and random oversampling are set by default, it can be determined that they have complexity $O(n_s^{\frac{3}{2}})$ [193]. Next, the base classifier training and prediction is taking some time in this algorithm. Four of them were used during the experimental evaluation – *Support Vector Machine* (SVM), *k-Nearest Neighbors* (KNN), *Gaussian Naïve Bayes* and *Decision Tree CART* (DTC).

SVM classifier has a complexity equal to $O(n_s^3)$ [1]. The complexity of the KNN Classifier and the *Gaussian Naïve Bayes* is $O(n_s d)$, where d is the number of dimensions [59]. The CART complexity is $O(dn_s^2)$ [238]. To summarize the different settings, it can be estimated that the highest complexity equal to $O(n_s^3)$ will be for the situation where the SVM model is used. In contrast, the lowest possible complexity will be equal to $O(n_s^{\frac{3}{2}})$ when random undersampling and random oversampling and the *GNB* base classifier are used.

4.1.1 Experimental evaluation

In this section, an experimental evaluation of the proposed method will be performed with a comparative analysis with selected *state-of-the-art* methods using imbalanced data streams with various characteristics.

RQ1: What is the best setup of clustering methods and cluster consistency metric in *DSC* method?

RQ2: What is the impact of the concept drift and the dynamic imbalance ratio data stream on *DSC*?

RQ3: How flexible is *DSC* on the imbalance ratio data stream?

RQ4: What is the predictive performance of *DSC* in comparison to the *state-of-the-art* methods?

Setup

All tests using a wide range of synthetic and real-world data streams were conducted according to the *test-then-train* experimental protocol [24]. This consists of processing the data stream as a data chunk and each of these data chunks is used first for testing and then for training the classification method. Exception is in the first data chunk, which is only used for learning. In this way, it is possible to test the method on the entire data stream without separating the test and training data. The results obtained are expressed in the form of metrics – *Recall*, $Specificity$, *Precision*, $F_1 score$ [235] and $Gmean_s$ [152]. Then the obtained scores were subjected to statistical analysis using Wilcoxon pair rank-sum tests. The study was performed using four base classifiers:

- *k-Nearest Neighbors* (*KNN*)
- *Support Vector Machine* (*SVM*)

- *Gaussian Naive Bayes* (*GNB*)
- *Decision Tree CART* (*DTC*)

The project was implemented in Python language using scikit-learn library. The implementation and the full set of results are publicly available on the Github repository[1]. The data streams were generated using MOA [24] and stream-learn [150] generators. Tab. 3.3 presents parameters such as a number of objects, chunk size, imbalance ratio, noise level or drift type of generated synthetic data streams. The proposed method was compared with selected *state-of-the-art* methods:

- *REA* – Recursive ensemble approach [51]
- *KMC* – K-means clustering undersampling ensemble [264]
- *L++CDS* – Learn++CDS [62]
- *L++NIE* – Learn++NIE [62]
- *OUSE* – Over and undersampling ensemble [89]
- *MLPC* – Multi-layer perceptron classifier

Experiment 1 - Parameter setup

The first experiment focused on determining the best setting of the parameters that affect the performance of the method. The parameters consist of three important components: the data undersampling method, the data oversampling method, and the balance parameter (BP), which determines at what imbalance ratio additional oversampling of the data should be performed. Given these three parameters, one best combination of settings will be determined and used in further experiments.

To begin, the best value for BP will be determined. The tests were performed on the value range from 0.05 to 0.45. The results obtained do not allow a clear verdict which setting is the most favorable (Fig. 4.2). It is noticeable that the $F_1 score$ and $Specificity$ obtain high results with a low value of BP equal to 0.05. This means that despite the large imbalance ratio that is in the data chunk, no additional oversampling is applied. The method avoids extra oversampling so that even though the data is imbalanced, it still fits the model. It is reasonable that this results in better scores that express the correctness of the majority class prediction. This unfortunately comes at a cost on the *Recall*. Therefore, a much better move would be to choose a value that gets significantly

[1]Repository link: https://github.com/w4k2/iot-ecml2019

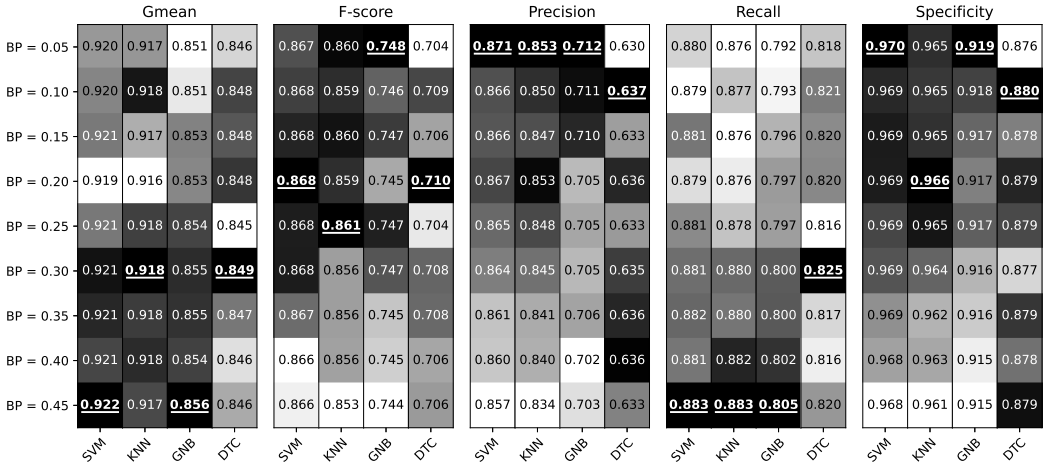


Figure 4.2: *Results of different balance parametr (BP) values. Darker is better, best value is bold and underscored*

better results for *Recall*. Such a value is BP equal to 0.45. This means that the method will attempt to balance the data, even if it deviates slightly from an even distribution of the data. This is also equivalent to a very good $Gmean_s$ and not much loss on the F_1score. This value will be used in further experiments.

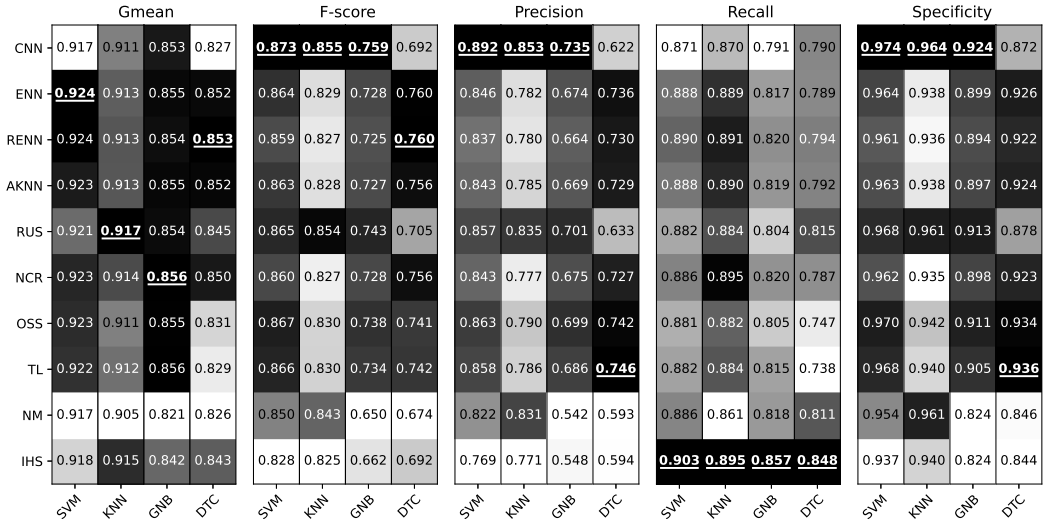


Figure 4.3: *Results of different undersampling methods. Darker is better, best value is bold and underscored*

Then, the best method for undersampling the data was determined. This included the following possible methods to choose from:

- Condensed Nearest Neighbors (CNN)
- Edited Nearest Neighbors (ENN)

- Repeated Edited Nearest Neighbors (RENN)
- All KNN (AllKNN)
- Instance Hardness Threshold (IHT)
- Near Miss (NM)
- Neighbourhood Cleaning Rule (NCR)
- One Sided Selection (OSS)
- Random Under Sampler (RUS)
- Tomek Links (TL)

For this experiment, choosing the best option is fairly straightforward (Fig. 4.3). The best results in the metrics $F_1 score$, $Precision$, $Specificity$ are obtained by the CNN method. The slightly weaker results are for the $Recall$, where the IHS method is the favorite. However, it is worth noting that the method of undersampling focuses on selecting a new subset of the majority class data, so it will be much more logical to rely on metrics that represent the quality of the majority class classification. Hence, the choice falls on the CNN method, which was used in further experiments.

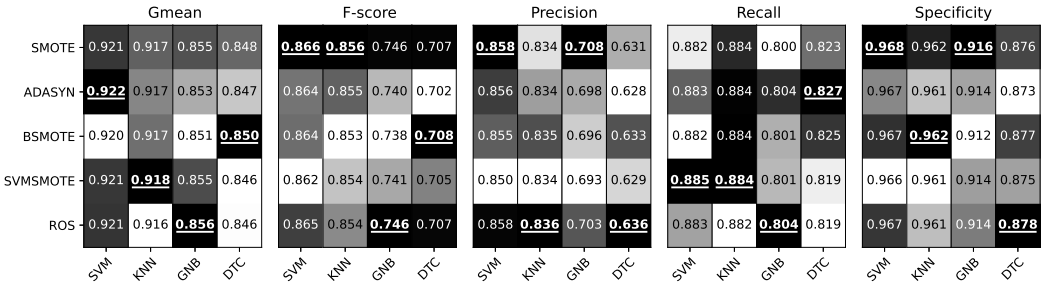


Figure 4.4: *Results of different oversampling methods. Darker is better, best value is bold and underscored*

The last parameter specified which method of oversampling the data would be used. The best method among the selected ones was searched for:

- ADASYN (ADASYN)
- Borderline SMOTE (BSMOTE)
- Random Over Sampler (ROS)
- SMOTE (SMOTE)

– SVMSMOTE (SVMSMOTE)

This experiment also does not allow one significant indication of one significantly better method (Fig. 4.4). There are very small and even hard to see differences between scores. Looking at the results more closely, it is noticeable that there are two variants that stand out a bit from the rest. One of them is the ROS method, which is the random amplification of samples. The results for this method are quite good for the *Precision*, it looks a little worse for *Recall*. The second favorite is the SMOTE method, which outperforms rest of the methods in the $F_1 score$. Since both of these variants have strongly similar results, for further research, the SMOTE method is chosen, which is based on a strategy of creating new samples rather than pure randomness like the ROS method.

Experiment 2 - Static imbalance

Having already prepared the best parameter setup, it is possible to proceed with proper experiments to verify the predictive performance of the proposed method. First, the reference methods with the *DSC* approach will be tested in the classification of data streams with a static imbalance ratio. This means that from the beginning to the end, the data stream will have the same distribution of the number of objects belonging to different classes.

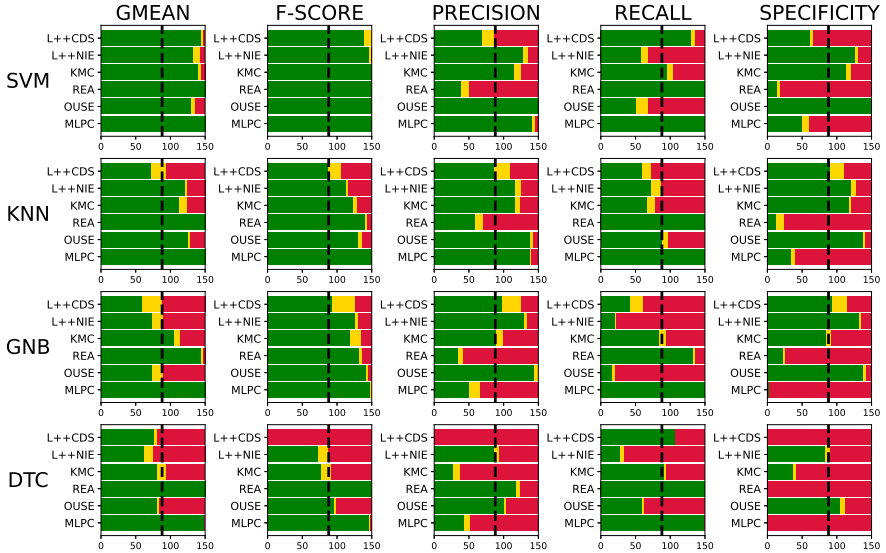


Figure 4.5: *Wilcoxon pair rank-sum tests for synthetic data streams with incremental concept drift. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*

Fig. 4.5 shows the results obtained for data with incremental concept drift. The *DSC* method obtains not the worst results. It can be seen that the highest results are obtained in the metrics $Gmean_s$ and F_1score which is manifested by a large number of statistically significant advantages with the other approaches. The weakest performance was for $Specificity$, where essentially half failed to achieve an advantage with statistical significance. In addition, it can be concluded that the *DSC* method performed best when using SVM or *KNN* base classifier.

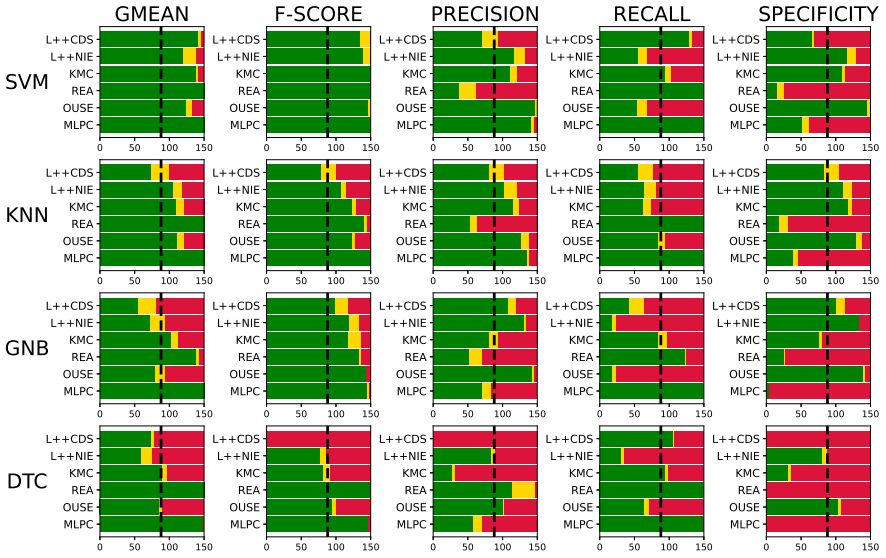


Figure 4.6: *Wilcoxon pair rank-sum tests for synthetic data streams with sudden concept drift. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*

For the data with sudden concept drift (Fig. 4.6), the results obtained are very similar to those of the previous experiment. It is hard to see any big differences, as there was a slight decrease in quality in some parts and an improvement in others. The *DSC* method is unlikely to show any special difference in classifying data streams with a particular type of drift, as it performs reasonably well in both variants.

Experiment 3 - Dynamic imbalance

The next experiments focus on testing the quality of methods that are subjected to classification of data with dynamic imbalance. This means that the imbalance ratio will change with each subsequent data chunk. This change will continue until, in about half of the data stream, the imbalance ratio reverses between classes and then returns to the initial value. The changing imbalance is typical for real origin data. It is worthwhile to test the proposed method against other selected *state-of-the-art* methods on this type of data.

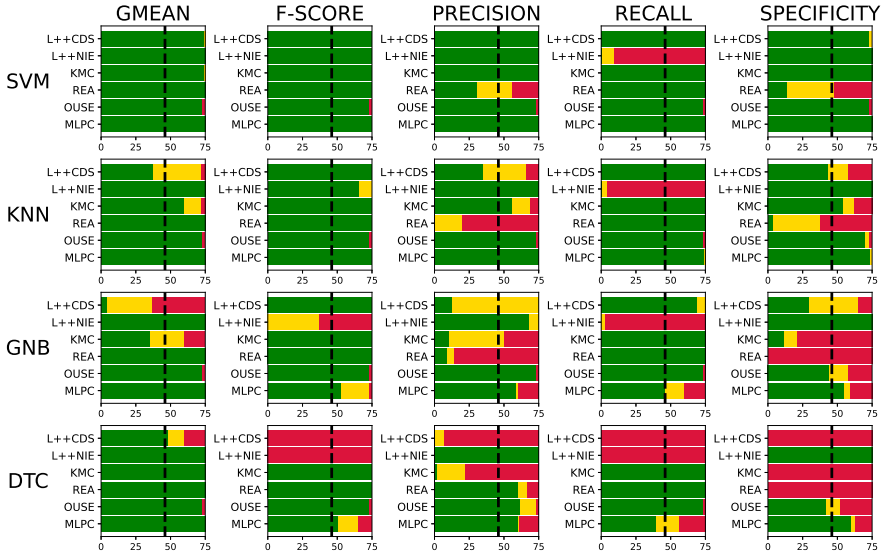


Figure 4.7: *Wilcoxon pair rank-sum tests for synthetic data streams with incremental concept drift and dynamic imbalance ratio. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*

The situation is slightly better for data with dynamic imbalance. The results in Fig. 4.7 show the result for data streams with incremental concept drift. The proposed method obtains very good results. In most of the pairwise comparisons performed, a superiority with statistical significance is obtained. When the *GNB* classifier and *DTC* are used, the results are slightly worse. *DSC* loses mainly with the Learn++CDS or Learn++NIE method. The best results are obtained using the SVM base classifier, where there are actually only wins except for one loss against the Learn++NIE method and two ties against the REA method. Looking at the results from the metrics side, the best scores are obtained for $Gmean_s$. Slightly worse but also high on the F_1score. The weakest results are obtained for $Precision$ and $Specificity$.

Similar results but slightly more favorable for the *DSC* method were obtained when testing synthetic data that have sudden concept drift (Fig. 4.8). Comparing this result and the previous one, it can be seen that there is a negligible improvement. However, it is sufficiently impressive that previously where *DSC* was on the border between a draw and a win it now obtains statistically better results. This applies mainly to the draws of the *DSC* method and the Learn++CDS method. The obtained results may indicate that the proposed method has outperformed Learn++CDS on data that has abrupt changes in feature distribution.

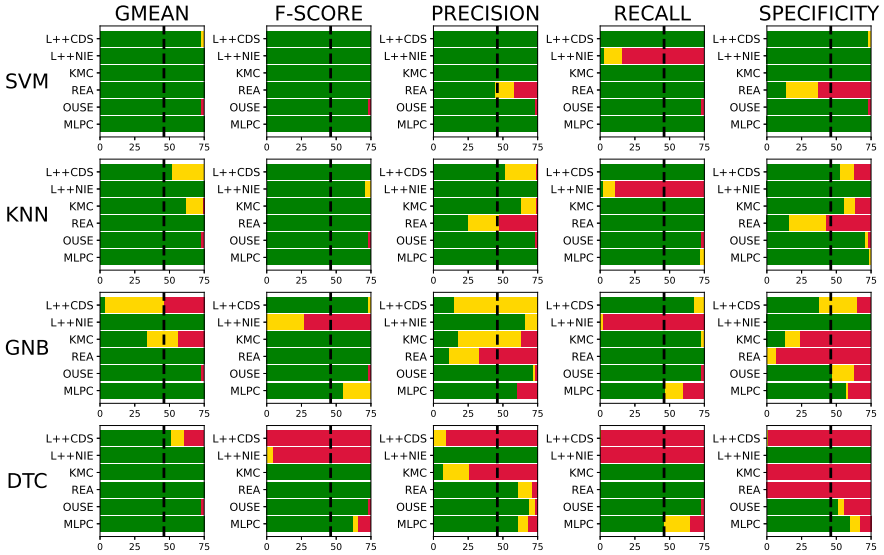


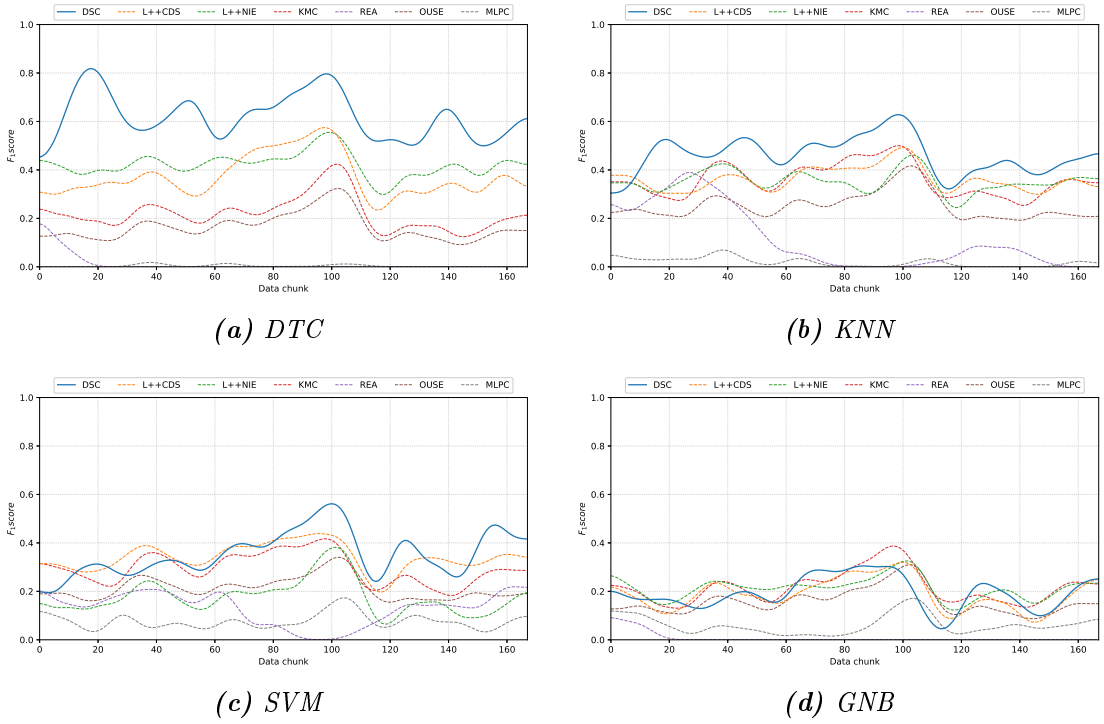
Figure 4.8: *Wilcoxon pair rank-sum tests for synthetic data streams with sudden concept drift and dynamic imbalance ratio. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*

Experiment 4 - Real data

The last experiment focuses on testing the performance of the proposed method on imbalanced real-world data streams. Unfortunately, this type of data is quite rare and difficult to obtain. Most often, one may find data streams that do not have a significant imbalance ratio or imbalanced data sets that are not affected by concept drift. There are various techniques for combining or generating artificial imbalance or drift, but they usually involve significant interference with the original data distribution. For this reason, this research will be limited to two real data streams [44], the characteristics of which can be seen in the table below (3.4):

First, the analysis of the results obtained will focus on the stream "poker-lsn-1-2vsAll". The plots (Fig. 4.10a–4.10d) show the runtime graph for each of the tested methods using four different base classifiers. These graphs represent the quality obtained by a given method in a given data chunk expressed by the $F_1 score$ metric. The results presented indicate that *DSC* performed best using the *DTC* and *KNN* base classifier. For these graphs, it is apparent at first glance that the *DSC* method has an advantage over the other methods. Worse results are obtained for SVM classifiers and the weakest for *GNB*. However, in both these results the performance is not the worst among the compared methods.

Looking at the scores obtained for the "covtypeNorm-1-2vsAll" data stream, most of the tested methods obtained highly similar performance. It is slightly noticeable that for the experiments using *DTC* classifier, the proposed method is in the lead competing with



(a) DTC *(b) KNN*

(c) SVM *(d) GNB*

Figure 4.9: *F_1score over the data chunks for data stream poker-lsn-1-2vsAll*

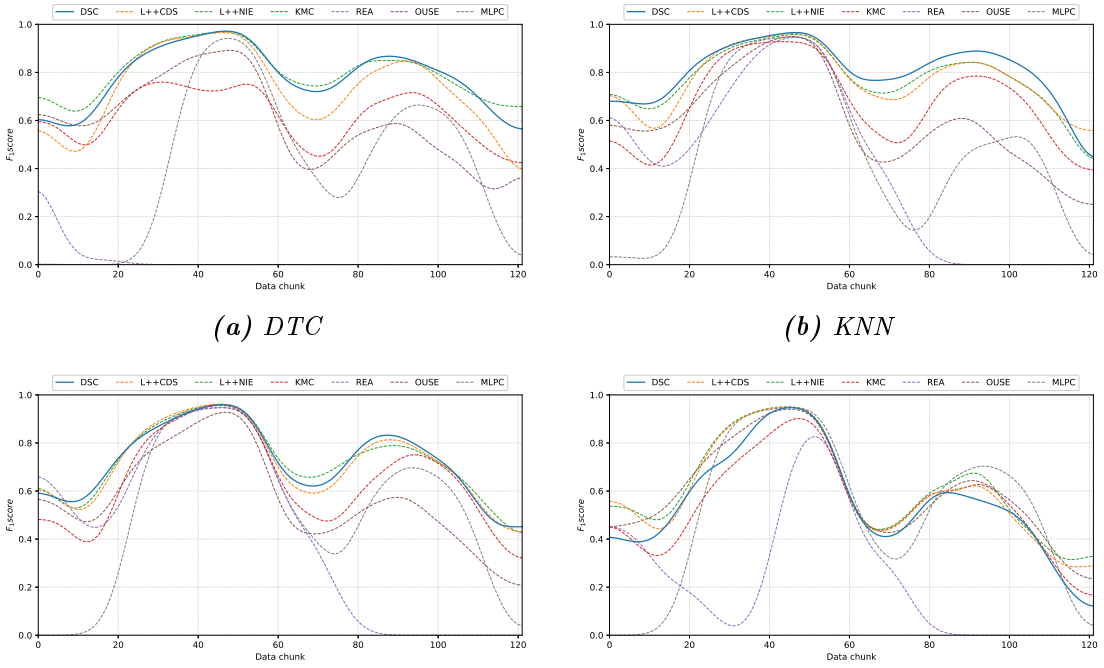
Learn++NIE method. A similar observation applies to the results for *KNN* and SVM, except that the Learn++CDS also is included in the top performers. The results of the *DSC* method using the *GNB* base classifier were the weakest, where it nevertheless performs at a very similar level to the other methods.

4.1.2 Lessons learned

To summarize the experimental evaluation conducted, the answers to the research questions posed earlier are presented below:

RQ1: What is the best setup of clustering methods and cluster consistency metric in DSC method?

The *DSC* method has a certain set of parameters and a wide range of possibilities to set them depending on the problem under consideration. It is quite difficult to determine one best setting, because it requires using appropriate strategies to get good results for a certain type of data. However, it was possible to propose a solution that can be treated as a default setting. The research and analysis of the obtained results showed that the best settings for the tested data streams are 0.45 for the balance parameter, SMOTE as oversampling and Condensed Nearest Neighbors as undersampling method.



(a) DTC *(b) KNN*

(c) SVM *(d) GNB*

Figure 4.10: *F_1score over the data chunks for data stream covtypeNorm-1-2vsAll*

RQ2: What is the impact of the concept drift and the dynamic imbalance ratio data stream on DSC?

The tests performed show that the proposed method has a good ability to deal with difficult data. This is valid for data streams that have concept drift as well as imbalance. By analyzing the results, it can be concluded that *DSC* performs much better for dynamic imbalance than for static imbalance. It means that the method can manage in much more difficult conditions. Moreover, it is worth noting that it achieves satisfactory and equally good results both when classifying streams with sudden and incremental drift.

RQ3: How flexible is DSC on the real data stream?

The results obtained on real data indicate the good classification ability of the proposed method on this type of data. It is worth noting that for some base classifiers *DSC* can achieve a significant advantage over other methods. Even in the worst case, where the obtained predictive performance does not outperform other methods, the results that *DSC* achieves are not significantly different from the others. The method has considerable potential for classifying imbalanced real-world data streams, which is a very promising feature.

RQ4: What is the predictive performance of DSC in comparison to the* state-of-the-art *methods?

The tests performed on a very wide set of synthetic data streams allowed for a fair comparison of the proposed method with other selected *state-of-the-art* approaches. The analysis of the obtained results allows to conclude that *DSC* obtains remarkably well results. In most of the comparisons that were performed using Wilcoxon pair rank-sum tests, the proposed method obtains a statistically significant advantage. Of course it is not an absolute advantage, where in each variant of the test *DSC* is better, but there is also no method that will always dominate.

4.2 Deterministic sampling classifier with weighted bagging

The **D**eterministic **S**ampling **C**lassifier with weighted **B**agging (*DSCB*) is a proposition of an extension existing **D**eterministic **S**ampling **C**lassifier (*DSC*) [28] (Section 4.1) classifier. It is a chunk-based method for imbalanced data stream classification. In the original approach, the main innovation was the use of a data accumulation technique to solve the imbalanced data classification problem. The new method expands on this idea by using a specially designed weighted bagging to create an ensemble from the accumulated data. In the rest of this section, the method will be described in much more detail.

Proposed method

The proposed approach as well as the original *DSC* method on which it is based, mainly focuses on the use of data accumulation techniques to deal with the imbalance data problem. A similar approach that involves collecting some of the patterns from previous data chunks can be found in other works [90]. However, it is worth noting that most methods only concentrate on the accumulation of data from the minority class. Then, by gathering these samples, one can easily oversample the incoming imbalanced data. Unfortunately, by omitting the majority class, the insight into the whole picture is lost. Increasing the number of available patterns in this way can very often lead to distorted decision boundaries. Hence, the innovation of the approach proposed in this work is the simultaneous accumulation of minority and majority data.

However, it should be kept in mind that this method is designed to solve the imbalance data classification problem. Storing the entire data chunks will not bring any benefit in terms of reducing the imbalance ratio. Therefore, this approach assumes a slightly different solution, which will ensure data continuity for both classes and at the same time will be a kind of compromise for data balancing. Each data chunk, before retaining it in data storage, goes through the undersampling process. This allows to reduce the number of majority class samples. Then such a data chunk can be stored entirely and used to enhance subsequent chunks. This method allows the imbalance ratio to be reduced along with preserving information about the relationship of both classes of objects. At the same time, it is worth noting that this approach solves another problem that may arise when storing only minority data. When the amount of minority data that is accumulated grows to a fairly large size that exceeds the imbalance of a single data chunk, a malfunctioning situation can occur where amplification of all data will lead to the imbalance toward the original minority class. It would require additional techniques to regulate the number of objects used and to determine which samples should be used for amplification. By

storing data from both classes, such a situation can be avoided because the class samples are amplified equally.

Despite using the method of data amplification for balancing a data chunk, it is worth noting that it does not necessarily ensure an even distribution of objects from the two classes. Especially, such a situation is much more likely at the initial stage of method activity, when the collected data has a very low size. Therefore, this idea is extended to an additional method that, after amplifying the data with real-life samples, performs additional amplification using artificially generated samples. Any method that implements the technique of oversampling imbalanced data can be used for this. This approach allows to balance the data chunk in a very easy way, even when the amount of collected data is insufficient. The final imbalance ratio in the data chunk that the algorithm is trying to achieve is determined by the parameter. This means that only enough synthetic patterns are generated to reach a certain threshold of imbalance. Additionally, it is worth noting that the oversampling and undersampling methods used in this method are treated as another parameter, which choice can be adjusted depending on the actual problem. Then such prepared data chunks can be used to build a classification model.

Primarily in the original *DSC* approach, the collected data at each iteration forms a new model built using a chosen base classifier. However, it is worth noting that the collected data along with the actual data chunk form a large set of patterns. Such a substantial dataset seems to be an ideal foundation for building the ensemble of classifiers. Therefore, the main innovation that has been proposed with respect to the *DSC* algorithm is the formation of classifier ensembles based on the bagging approach. Using such a technique will simultaneously increase the classification ability of the method and maintain a better generalization.

Considering the problem of data stream classification, it is worth noting that the more recent the data, the potentially much more valuable it is from a model perspective. Samples stored from previous data chunks were saved at different times. This means that some patterns are much older and others are fairly new. Such information makes it possible to assign values to determine the age of each sample. The basic version of bagging is based on creating certain subsets of data, which consist of drawn patterns with uniform distribution. The proposition that was used in this algorithm tries to use the information about the age of the samples when creating the bagging subsets. Hence, a decay function is proposed that determines the chance of drawing a given sample using the following formula (Eq. 4.1):

$$w(x_i) = 2\pi e^{-\frac{(t-k)\gamma}{2}} \tag{4.1}$$

where x_i is the $i-th$ pattern from a $k-th$ stored data chunk, t is a index of current data chunk, and γ is a user parameter that allows to adjust the strength of weight degradation over time. This means that the older the sample is, the much smaller the weights are, which affects the chance of being drawn. Some heuristics have been introduced into the proposed weight function, which at the beginning of the stream increases the chance of drawing samples from the minority class. This means that for the first few iterations, the weights for the minority class are calculated in a slightly different way according to the formula (Eq. 4.2):

$$w(x_i) = \begin{cases} 2\pi e^{-\frac{(t-k-L)\gamma}{2}} & \text{if } t < d_h \text{ and } x_i \text{ is minority class sample} \\ 2\pi e^{-\frac{(t-k)\gamma}{2}} & \text{otherwise} \end{cases} \tag{4.2}$$

where L is a parameter that determines the intensity, t is an index of current data chunk and d_h is a parameter that defines how long this heuristic should last. Such a procedure helps to improve the performance of the classification, which is usually very low due to the poor representation of minority data. Then from these weights, the probability of each sample being drawn must be calculated, which is done according to the formula (Eq. 4.3):

$$P_b(x_i) = \frac{w(x_i)}{\sum_{n=1}^{N} w(x_i)} \tag{4.3}$$

where $w(x_i)$ is the weight that translates to the chance of drawing a given pattern, and N_s is the number of total samples contained in the dataset. Then, after drawing several bagging sets, new models are built using the base classifier to form ensembles of classifiers. The number of models that will be created and at the same time the number of data sets that will be drawn is determined by the parameter s. The final decision that is made by the method is based on the support accumulation rule that is returned by the individual models that form the ensemble.

Algorithm

To fully and correctly demonstrate how the proposed method works, it is necessary to present in pseudocode a loop that simulates the arrival of subsequent data chunks of the stream. This allows us to specify the t-th element of the method, such as models, stored samples, etc. However, this is already a part belonging to the evaluation and not to the method itself. It is known that after each training of the base classifiers on a new data chunk, the whole ensemble is returned, which as a trained model is used for prediction.

Unfortunately, the return statement could imply that the algorithm is interrupted at this point and the data stream will not be fully processed. Therefore, it is a simplification to assume that the ensemble is returned after each data chunk.

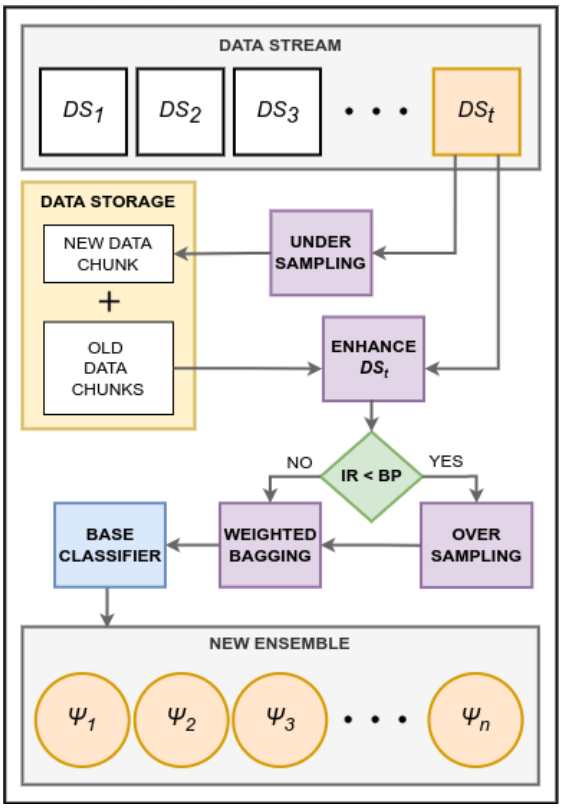


Figure 4.11: *Deterministic Sampling Classifier with weighted Bagging*

The main idea of the proposed method is represented graphically by a diagram 4.11. A detailed description of the entire learning procedure will be presented to understand better the idea and how the proposed method is working (Alg. 4). As an input, the algorithm takes several elements that are necessary for the further stage. First, a data stream is denoted as DS, which is passed to the algorithm in data chunks (DS_t). The base classifier learning method (BCL) and the method for oversampling (OSM) and undersampling (USM) must also be specified. In addition, it is required to indicate the maximum ensemble size (s), the maximum number of stored data chunks (S), and to set a value for the balance parameter (BP).

In the first step, the method clears the ensemble of classifiers. During the first iteration, when t is equal to 1, the ensemble is already empty. However, subsequent iterations will fill it with models that should be removed from the ensemble with each new data chunk. Then the undersampling of the current data chunk (DS_t) is performed using the USM method. The result of this operation is written to DSU_t. Later, when the Accumulated Data Storage (ADS), the storage of the collected data, is not empty, the

DS_t is amplified using the patterns collected there. The next step is to determine if DS_t after amplification has an imbalance ratio (IR) above a predetermined threshold (BP). When this condition is met and the data is still too imbalanced, then oversampling is performed, which is the generation of artificial patterns using the OSM method. This step is usually performed on few first data chunks when the method does not gather enough data.

Then, having the data set already prepared, the procedure of learning new models begins. The new models make up the ensemble of classifiers, which are built using the weighted bagging technique. First, the weights for each sample must be determined according to the Eq. 4.2. After that, these weights are transformed into an array of probabilities of drawing a given sample according to the Eq. 4.3. The created probabilities are stored in an array P_t. Then, m times the process of learning a new model is performed. According to the predetermined probabilities, samples are drawn with replacement and stored as a subset of data – B_t. In the next step, these data are used to build a model (Ψ_t) based on the selected base classifier learning method (BCL). The newly created model is added to the pool of classifiers (Π).

The next step is to write to ADS the current data chunk (DSU_t) that was previously balanced using the undersampling method. An essential assumption is to perform this step now. Writing the data to ADS too early could result in a situation where some objects will be unnecessarily duplicated during DS_t amplification. When the size of ADS exceeds a predetermined threshold for the maximum number of saved chunks (S), the oldest data chunk (DSU_{t-S}) is removed from it. Finally, the method returns the created pool of Π classifiers.

Computational complexity analysis

The computational complexity of the proposed method consists of several components. First, the chosen base classifier has a significant influence. Four different classifiers were used during the experiments – *Support Vector Machine* (SVM), *k-Nearest Neighbors* (KNN), *Gaussian Naïve Bayes* (GNB) and *Decision Tree CART* (DTC). Let us assume that n_s denotes the number of objects in one data chunk. SVM has a complexity of $O(n_s^3)$ [1]. The complexity of the KNN Classifier and the *Gaussian Naïve Bayes* is equal to $O(n_s d)$, where d is the number of dimensions [59]. The CART complexity is $O(n_s^2 d)$ [238]. Then the complexity is affected by the chosen method for undersampling and oversampling. Assuming that random undersampling and random oversampling are set by default, it can be determined that they have complexity $O(n_s^{\frac{3}{2}})$ [193]. Next, it is important to estimate the complexity associated with accumulated data storage. There

Algorithm 4 DSCB

Input:

DS – data stream
DS_t – t-th data chunk of data stream DS
BCL – base classifier learning method
OSM – oversampling method
USM – undersampling method
BP – balance parameter
s – maximum size of classifier ensemble
S – maximum size of stored data chunks

Symbols:

ADS – accumulated data storage
DSU_t – undersampled t-th data chunk
B_t – bootstraped samples from t-th data chunk
P_t – probability of selecting samples from t-th data chunk
Ψ_t – t-th model of ensemble

Output:

Π^t – final ensemble for t-th data chunk

for $t = 1, 2, \ldots$ **do**
  $\Pi \leftarrow \emptyset$
  $DSU_t \leftarrow$ Undersample DS_t using USM
  if $ADS \neq \emptyset$ **then**
    $DS_t \leftarrow DS_t \cup ADS$
  end if
  if IR of $DS_t > BP$ **then**
    $DS_t \leftarrow$ Oversample DS_t using OSM
  end if
  $P_t \leftarrow$ Determine the probability of samples according to Eq. 4.3
  for $i = 1$ to s **do**
    $B_t \leftarrow$ Bootstrap samples from DS_t according to P_t
    $\Psi_t \leftarrow$ Train model on B_t using BCL
    $\Pi \leftarrow \Pi \cup \Psi_t$
  end for
  $ADS \leftarrow ADS \cup DSU_t$
  if $|ADS| > S$ **then**
    $ADS \leftarrow ADS \setminus DSU_{t-S}$
  end if
end for

are only two operations performed on ADS – writing and reading. It takes $O(1)$ time to add a new item to the list of any size [54]. When saving new data, the time complexity can not exceed $O(n_s)$. Reading data from ADS and adding it to the current training data chunk does not exceed the complexity of $O(Sn_s)$, where S is the maximum number of chunks stored in the memory. Finally, another component that affects the aggregate complexity of the method is bagging, whose complexity is $O(sn_s log(n_s))$ [103], where s is the maximum number of classifiers. The proposed method has many different parameters that have a significant impact on the final time complexity. The plethora of different settings do not allow for a clear determination of the time complexity. However, if one assumes that the SVM is chosen as the base classifier, then the whole method's complexity would be $O(n_s^3)$ because it is one of the highest complexities. Choosing the base classifier *Gaussian Naïve Bayes*, the complexity will be $O(n_s^{\frac{3}{2}})$, which is due to the complexity of the undersampling or oversampling method. This complexity should be considered as one of the scenarios. Other more sophisticated data sampling methods usually have higher complexity.

4.2.1 Experimental evaluation

The conducted experiments were designed to test the newly proposed *DSCB* method in practice. The research focused on two main aspects – finding appropriate settings for the method and comparative analysis using data streams with different characteristics. The following research questions were posed:

RQ1: What is the best parameter setup for *DSCB*?

RQ2: How robust is the proposed method to data streams with various imbalance ratios?

RQ3: How resilient is *DSCB* to label noise on different levels?

RQ4: What is the predictive performance of *DSCB* for dynamic imbalance ratio data stream?

RQ5: What is the proposed method's predictive performance in a comparison to the *state-of-the-art* classifiers?

Setup

Experimental evaluation was performed according to the *test-then-train* manner [24] approach. This means that each data chunk was firstly used for testing the model and then for training. The exception is the first data chunk, which was omitted from the

testing phase. To evaluate the classification quality obtained by the methods, the following selected metrics were used: *Recall*, *Specificity*, *Precision*, $F_1 score$ [235] and $Gmean_s$ [152]. Then the obtained scores were subjected to statistical analysis using Wilcoxon pair rank-sum tests. The proposed method was compared with selected imbalanced data stream classification *state-of-the-art* algorithms:

- *REA* – Recursive ensemble approach [51]
- *KMC* – K-means clustering undersampling ensemble [264]
- *L++CDS* – Learn++CDS [62]
- *L++NIE* – Learn++NIE [62]
- *OUSE* – Over and under-sampling ensemble [89]
- *MLPC* – Multi-layer perceptron classifier

The implementation of the proposed method along with the experimental environment was programmed in Python. The project implementations with results are available on a public Github repository[2]. Additionally, the environment uses a few ready-made implementations of the base models from the scikit-learn library [200]. Finally, four selected base classifiers were used in all experiments:

- *k-Nearest Neighbors* (KNN)
- *Support Vector Machine* (SVM)
- *Gaussian Naïve Bayes* (GNB)
- *Decision Tree CART* (DTC)

Three different sets of data streams were used to conduct the experiments – real data streams and synthetic data streams from two generators. The most important is real data, because it allows testing methods on data streams that at least try to mimic the difficulty of real-life classification problems. In addition, the data were supplemented by a large set of synthetically generated streams, which were provided by two data generators – MOA [24] and stream-learn [150]. Tab. 3.3 presents parameters such as a number of objects, chunk size, imbalance ratio, noise level or drift type of generated synthetic data streams.

[2]Repository link: https://github.com/w4k2/dscb

It is also worth describing how the results will be presented. For real data streams (Tab. 3.4), it is possible to plot runs and analyze the predictive performance expressed by the chosen metric. The problem arises during the analysis of synthetic data because their number is significant. This necessitates the use of a suitable technique to aggregate the results without much loss of information. The Wilcoxon rank-sum statistical test was chosen for analysis and pairwise comparison with other selected *state-of-the-art* methods. Then, depending on whether the result indicates a win, a draw, or a loss, it is indicated by the corresponding color on the graph (green, yellow, or red) (Fig. 4.18–4.21). In addition, there is a dashed vertical line on the graph, which is a critical value with a significance level of 0.05. Crossing this line with a green bar means that the *DSCB* method is better with statistical significance. In the figures, the results are grouped by metrics (columns) and base classifiers (rows).

Experiment 1 - Parameter setup

One of the first experiments that was conducted was aimed at finding the best possible parameter setup for the proposed method. Because of the wide range of parameters, the influence of only selected parameters on the predictive performance of the *DSCB* method was investigated. The parameter values were chosen in a global way. It means that only one setting was searched for, which will be used in all evaluations regardless of changing data streams. For this study, a certain set of synthetic streams was used that had similar characteristics to the streams used in the rest of the experiments. These data had 10 features with numerical values, 100000 samples, 10 attributes, 10%, 20% or 30% minority class samples and 0% or 5% label noise. However, to avoid overfitting the proposed method, these data streams are not repeatedly used in any subsequent experiment. That is, they were used only once to search for the best method settings. The obtained results are presented in the form of figures. In each figure, the medians were calculated using the obtained metrics values for each stream separately. Then the average for all streams with one base classifier was calculated. Results from this experiment are grouped by the base classifier and metric.

First, it was investigated how γ parameter affects the performance. γ is one of the components in the formula that determines the weight of the samples (Eq. 4.1) in the bagging. This parameter is responsible for how fast the degradation of the stored data chunks progresses over time. At first glance, it is apparent (Fig. 4.12) that depending on how much γ decreases, the resulting metrics increase their scores proportionately. The exception is *Recall*, which shows the opposite trend. However, it is worth noting that the differences for the *Recall* between the worst and the best score are around 20%. For the *Precision*, this difference is already significantly larger. Since the aggregate metrics

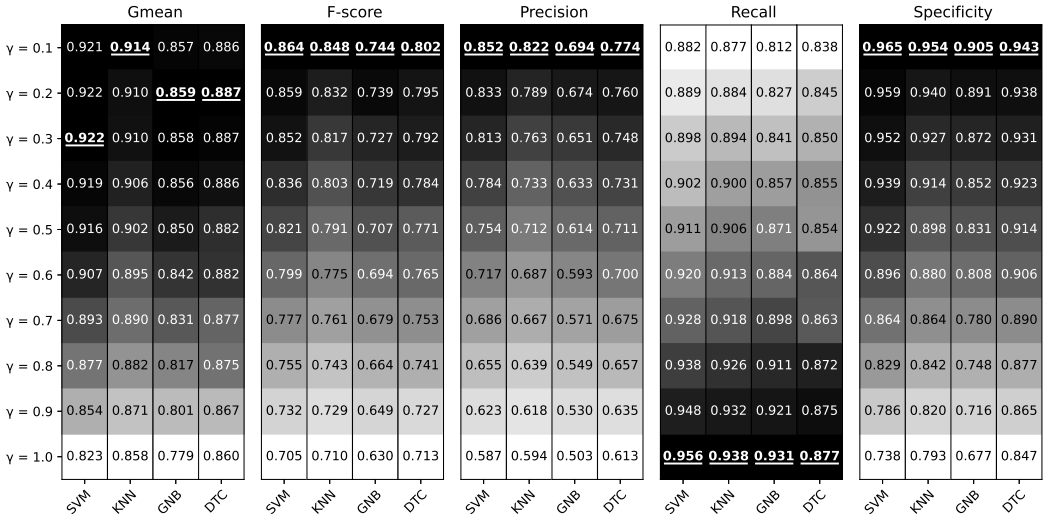


Figure 4.12: *Results of different γ values. Darker is better, best value is bold and underscored*

($Gmean_s$ and F_1score), obtain the best scores at $\gamma = 0.1$, this value is set as the default for subsequent experiments.

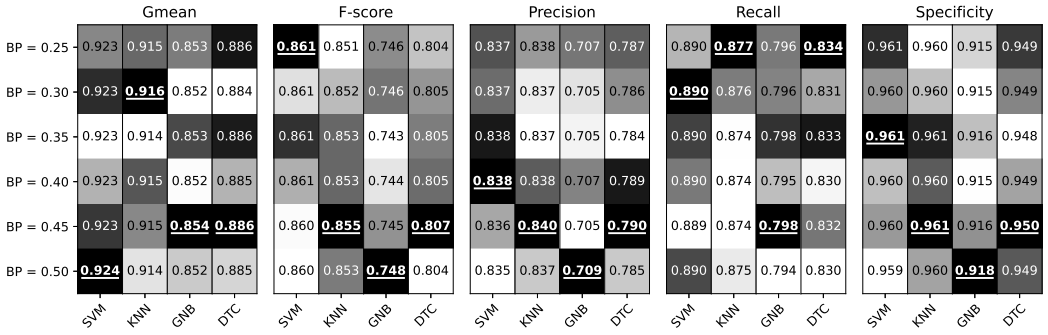


Figure 4.13: *Results of different balance parametr (BP) values. Darker is better, best value is bold and underscored*

Then the best setting was searched for the parameter controlling the final imbalance ratio after oversampling the data chunk. The results presentend on Fig.4.13 do not make it easy and unambiguous to determine the best option. It is an interesting observation that 50% ratio, which is an equal distribution of the class samples, does not allow to obtain significantly the best predictive performance. However, after analysis, the value equal 45% was selected. The results for this setup were usually with the highest score. This means that the method will not strive to perfectly balance the data, but will maintain a very slight imbalance.

The next experiment is to select the best techniques for undersampling and oversampling. First, the method for oversampling was searched. Five different approaches were

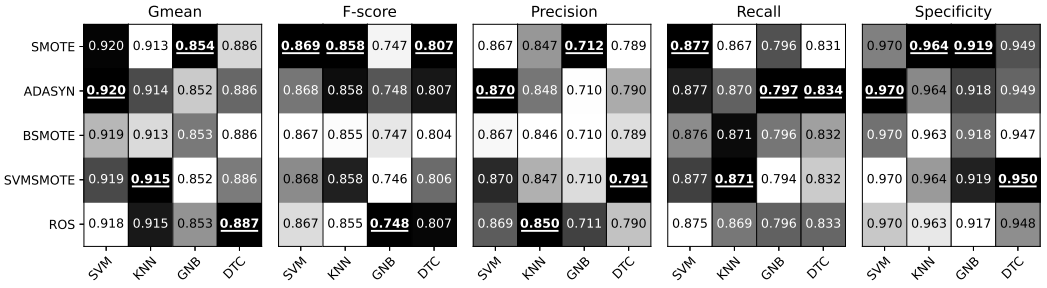


Figure 4.14: *Results of different oversampling methods. Darker is better, best value is bold and underscored*

proposed as a pool of available solutions. Below is a list of them along with the names and acronyms used in the presented results:

- ADASYN (ADASYN)
- Borderline SMOTE (BSMOTE)
- Random Over Sampler (ROS)
- SMOTE (SMOTE)
- SVMSMOTE (SVMSMOTE)

The obtained scores presented on Fig. 4.14 do not allow for a quick and unambiguous selection of the best method. Differences between them are very subtle and usually do not exceed one hundredth of score. However, a much more in-depth analysis reveals that most often the best score is obtained by the SMOTE method. This advantage is noticeable for most metrics and base classifiers. Based on such observations, the SMOTE method is selected for further experiments as the default oversampling setting.

Then different variants of the undersampling techniques were analyzed. The best method is sought among a 10 selected approaches. Below is the list of them along with the names and acronyms used in the results:

- Condensed Nearest Neighbors (CNN)
- Edited Nearest Neighbors (ENN)
- Repeated Edited Nearest Neighbors (RENN)
- All KNN (AllKNN)
- Instance Hardness Threshold (IHT)

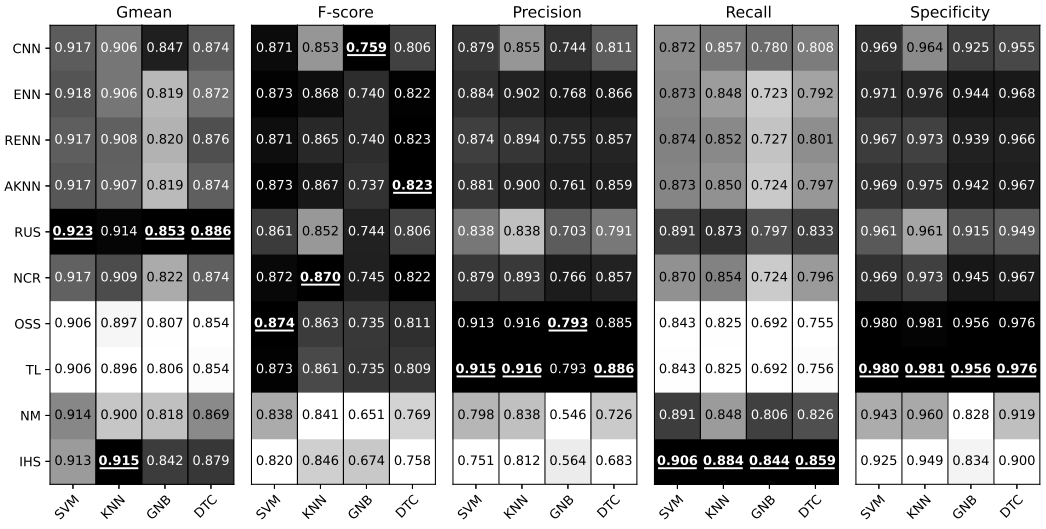


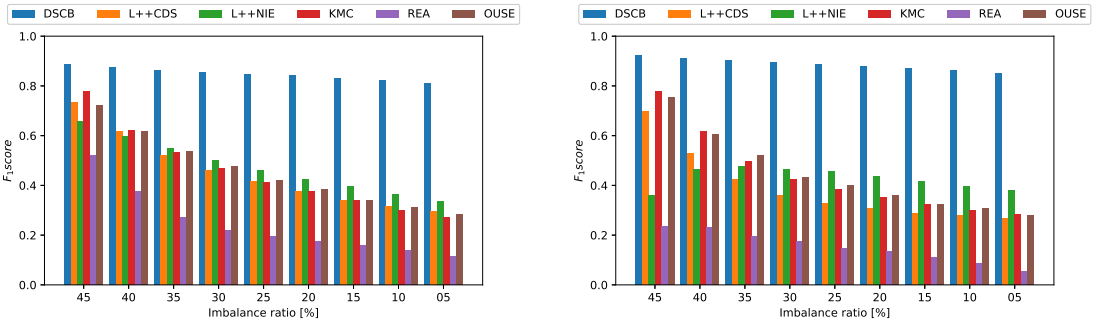
Figure 4.15: *Results of different undersampling methods. Darker is better, best value is bold and underscored*

- Near Miss (NM)
- Neighbourhood Cleaning Rule (NCR)
- One Sided Selection (OSS)
- Random Under Sampler (RUS)
- Tomek Links (TL)

As in the previous experiment, clearly identifying the best setup is not an easy task. The differences in the obtained results between the different variants have slightly larger values (Fig. 4.15). One method that achieves very good results is undersampling using Tomek Links (TL) technique. Unfortunately, the values of *Recall* for this method are on the lowest level among others. The best result in the *Recall* is obtained by the Instance Hardness Threshold (IHT) method, but again $Specificity$ is very poor for this method. The safest compromise between the *Recall* and $Specificity$ is obtained by the Random Under Sampler (RUS) method. Furthermore, this method gets the best results for $Gmean_s$. Due to the above, it is chosen as among other undersampling techniques and will be used in further experiments.

Experiment 2 - Label noise and imbalance ratio

Next, two experiments were designed to check how the proposed method performs on data streams with varying imbalance ratio and label noise. To ensure uniform conditions and eliminate factors that could adversely affect the final analysis result, a specially



(a) GNB *(b) SVM*

Figure 4.16: *Selected mean results from different imbalance ratio*

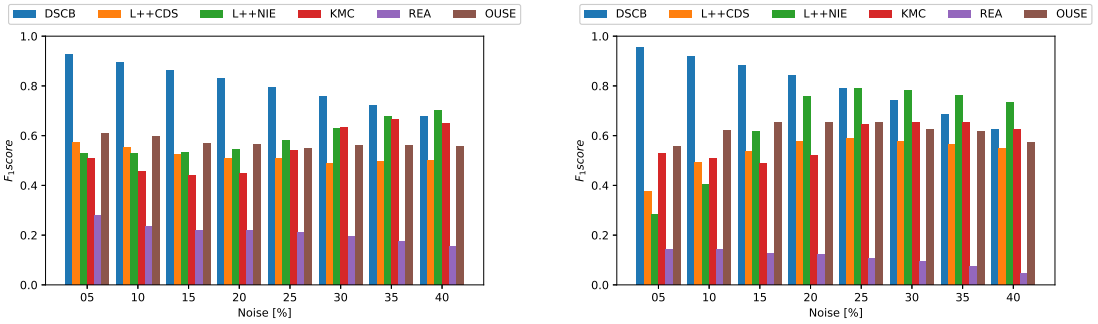
prepared set of imbalanced data streams with no concept drift was generated. Each of them consists of 10000 objects, 10 attributes with numerical values, and has 5 different variants of generator randomness.

The first experiment focused on testing the effect of the methods on predictive performance with varying imbalance ratios. The imbalanced level is denoted as the percentage of minority class samples in the entire data stream. The prepared data streams had values ranging from 45% to 5%. The obtained results (Fig. 4.16a, Fig. 4.16b), clearly indicate that the proposed method does not show a very negative effect on the changing level of imbalance. Some decreasing trend is observed. However, these results are not very different from each other. Compared with other methods, *DSCB* performs very well in this experiment.

The next experiment had similar data conditions as the previous one. However, it focused on testing the effect of the label noise present on the classification quality of the methods. This noise level represents the percentage of the data whose labels were swapped with the opposite class. In other words, when the data stream has 50% noise, then half of the labels describing objects have inverted class labels. For an in-depth study, data streams having label noise from 5% to 40% were generated. The presented results (Fig. 4.17a, 4.17b) show that increasing noise levels have a negative effect on the performance of all methods. This negative effect applies slightly weaker to *DSCB*. For the data stream that has 40%, the proposed method achieves a quality equal to 0.7 in the $F_1 score$. However, it is worth noting that a high label noise is a much more difficult classification problem.

Experiment 3 - Static imbalance

The main set of experiments focuses on the overall evaluation of the proposed approach compared to other selected *state-of-the-art* methods. The conducted tests will be divided



(a) GNB *(b) SVM*

Figure 4.17: *Selected mean results from different label noise*

into a few subsections grouped by the type of used data.

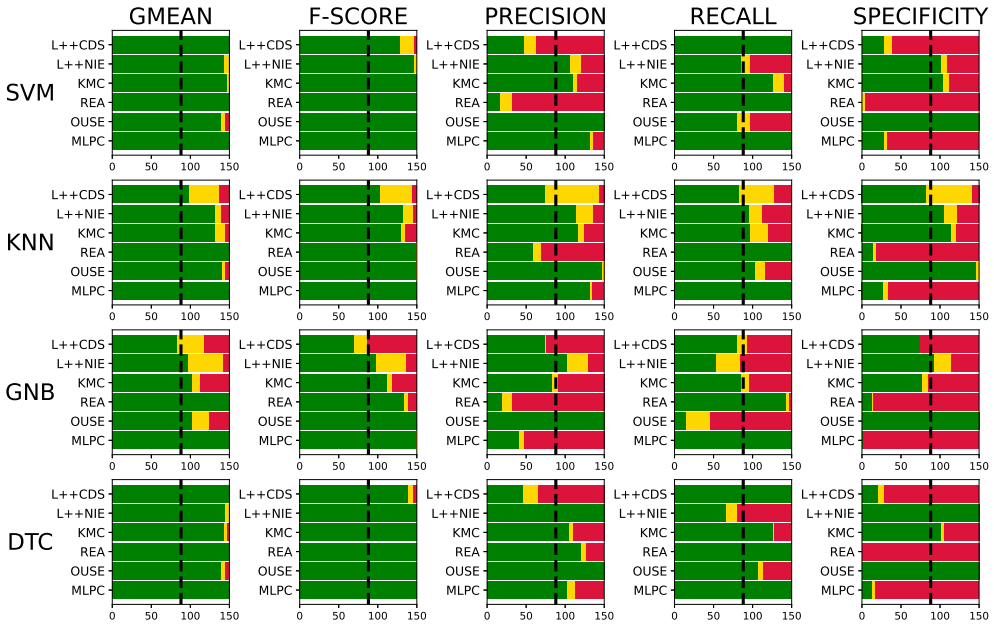


Figure 4.18: *Wilcoxon pair rank-sum tests for synthetic data streams with incremental concept drift. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*

The next experiment uses synthetic data streams with a static imbalance ratio. The obtained results presented on Fig. 4.18 contain a data stream with incremental concept drift. At first glance, the proposed method results look quite good since most of the comparisons are winning. The *DSCB* method achieves the best score on aggregated metrics. Compared to rest of the methods, it gets a statistical advantage or is on the verge of a tie with scores from $Gmean_s$ and F_1score. *DSCB* also performs well in the *Recall* but unfortunately sometimes is slightly worse for *Precision* and *Specificity*. Analyzing the results from the base classifiers' point of view, the *DSCB* method in

combination with the *KNN* classifier performs best. Very similar results are achieved for streams with sudden drift, which can be seen in Fig. 4.19.

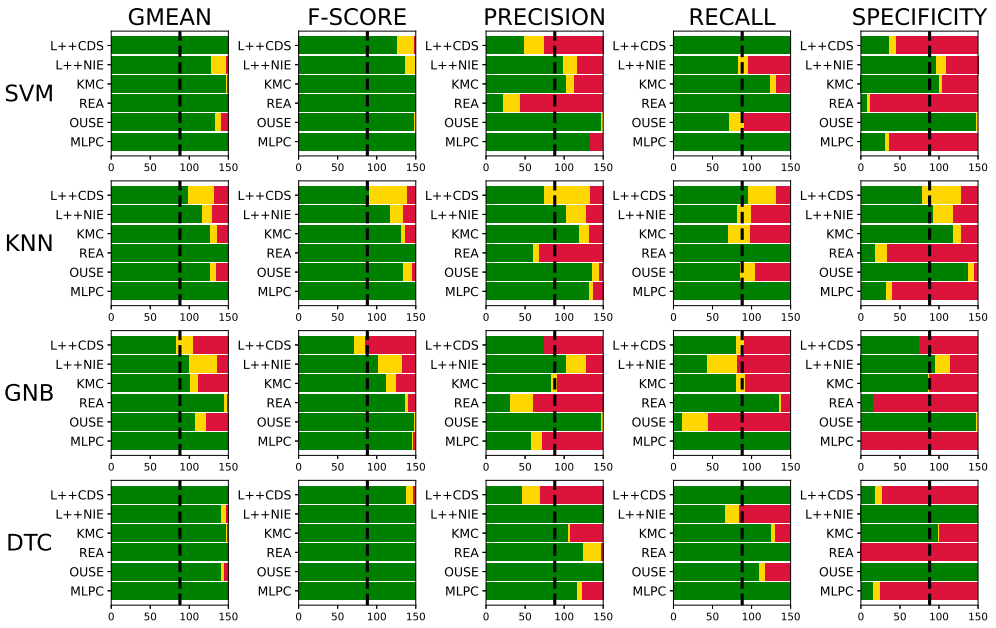


Figure 4.19: *Wilcoxon pair rank-sum tests for synthetic data streams with sudden concept drift. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*

Experiment 4 - Dynamic imbalance

Another experiment was designed to test the *DSCB* method when classifying data streams with dynamically changing imbalance ratios. The dynamic imbalance is a characteristic of real-world data. It is helpful to perform tests on data that attempt to simulate such a phenomenon. Moreover, this can be taken as some factor that increases the difficulty of the classification problem.

However, despite the increased difficulty, the *DSCB* method obtains much better results than in the previous experiment. In most comparisons that are shown in Fig. 4.20 with incremental concept drift, the proposed method obtains a statistically significant advantage. Learn++NIE is unique in that it achieves better scores for the *Recall* than the *DSCB* method. This is observed for all variants of the base classifier. Very similar results are obtained by *DSCB* for data with dynamic imbalance ratio and sudden concept drift, as shown in Fig. 4.21. Furthermore, there is an advantage with statistical significance in most of the comparisons made in this experiment.

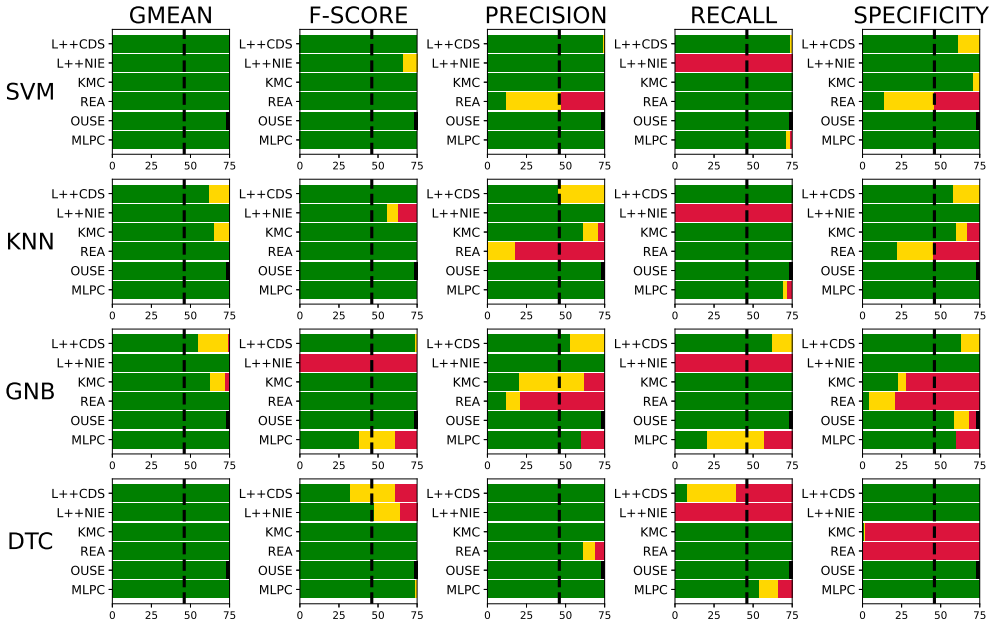


Figure 4.20: *Wilcoxon pair rank-sum tests for synthetic data streams with dynamic imbalance ratio and incremental concept drift. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*

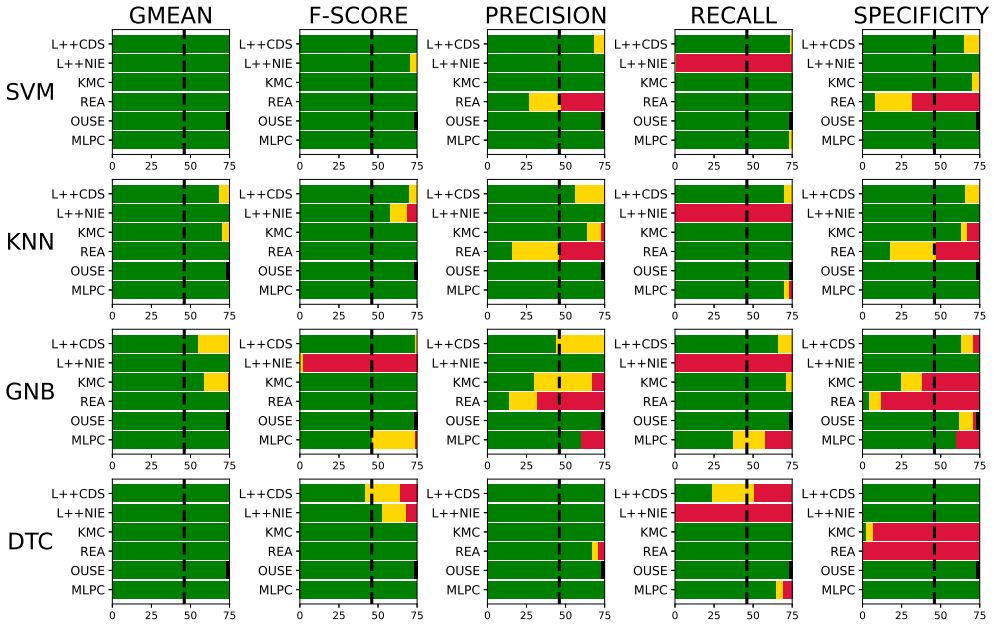
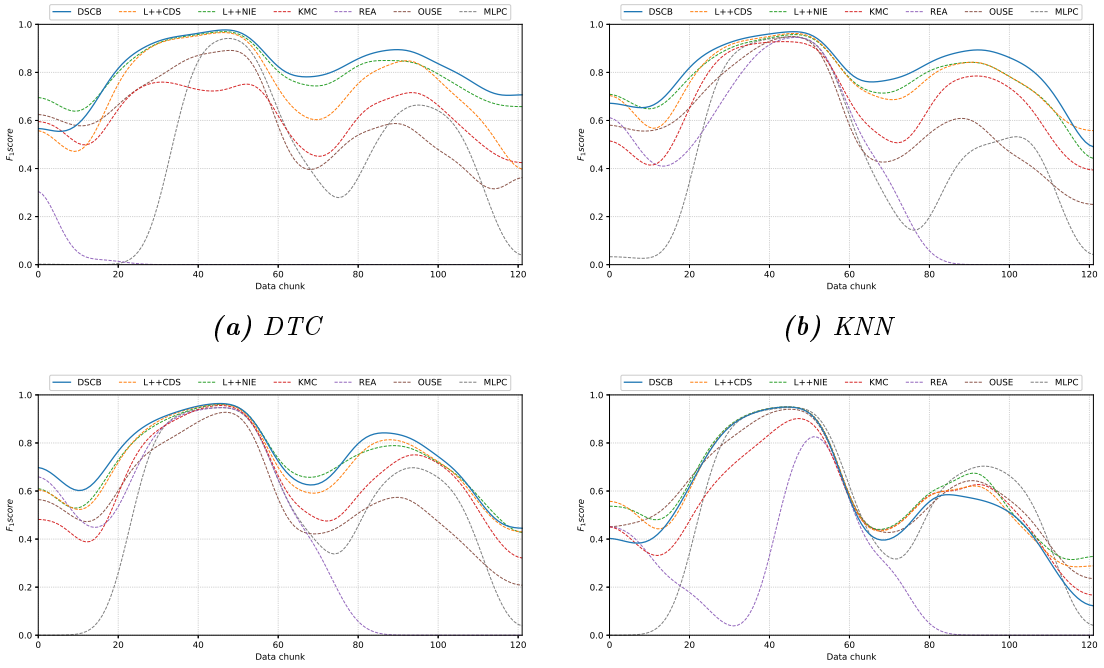


Figure 4.21: *Wilcoxon pair rank-sum tests for synthetic data streams with dynamic imbalance ratio and sudden concept drift. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*

Experiment 5 - Real data

The last experiment examines the predictive performance of *DSCB* on real data streams. It is worth noting here a particular problem with this type of data. Real origin data very



(a) DTC *(b) KNN*

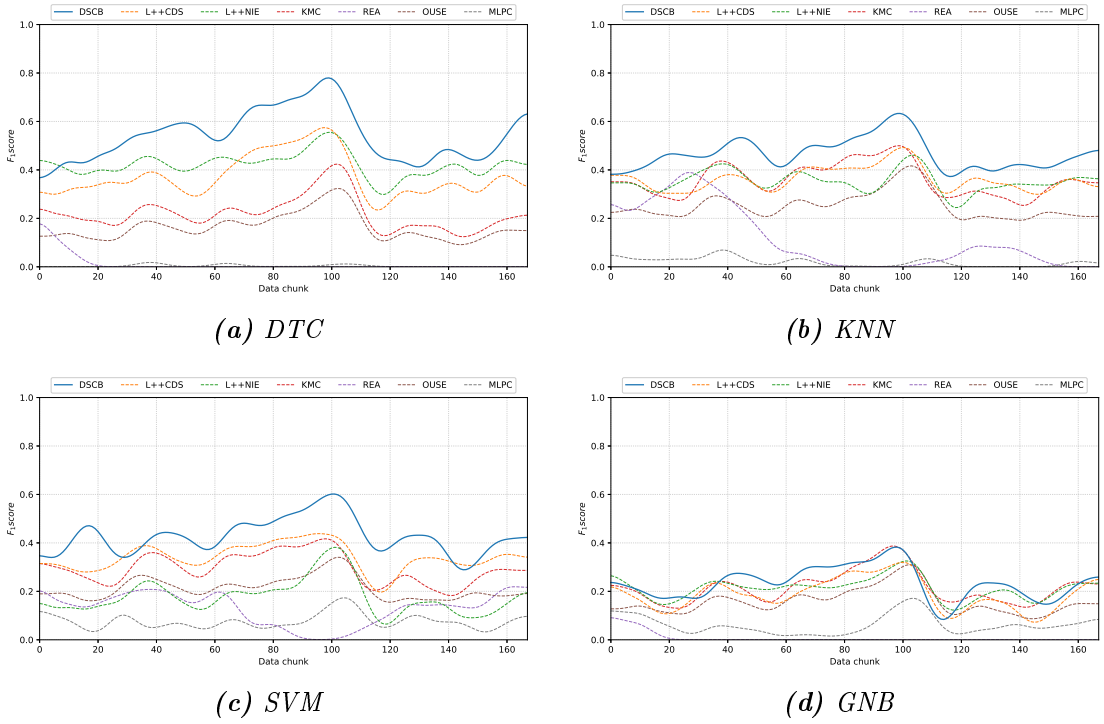
(c) SVM *(d) GNB*

Figure 4.22: *F_1score over the data chunks for data stream covtypeNorm-1-2vsAll*

often does not combine these two issues. This means that it is easy to find data with uneven class distribution, but it is usually a dataset with nothing in common with the data stream. On the other hand, one can find many data streams with different types of drifts, but most often, they do not have a significant imbalance ratio. Therefore, only two data streams (Tab. 3.4) that satisfy the above requirements [44] were selected. Since only two streams of real data were used in this research, it is possible to analyze the results using waveform graphs. These graphs show the quality of method classification expressed by metrics in a given data chunk. The abscissa axis describes the data chunk index and the ordinate axis describes the metrics scores.

First, the obtained results on the stream "covtypeNorm-1-2vsAll" using four different base classifiers will be analyzed. It is expressed by the F_1score and presented in Fig. 4.22a–4.22d. It is noticeable that the *DSCB* obtains high predictive performance. Using base classifiers *DTC* (Fig. 4.22a) and *KNN* (Fig. 4.22b), the metric initially equals about 0.6. Then it increases to about 0.9 and remains at that level. Compared with the other methods, *DSCB* usually achieves the best results. It performs worst for the *GNB* base classifier (Fig. 4.22d). However, it does not deviate significantly from the other tested methods.

The next chosen stream was "poker-lsn-1-2vsAll". The results expressed by the F_1score are shown in Fig. 4.23a–4.23d. Here the situation is as clear as in the previous experiment.



(a) DTC *(b) KNN*

(c) SVM *(d) GNB*

Figure 4.23: *F_1score over the data chunks for data stream poker-lsn-1-2vsAll*

The advantage of *DSCB* over the other methods is superior when using *DTC*, *KNN*, or *SVM* base classifier. An interesting observation is given by comparing the best results obtained over the entire data stream. It can be seen that for the base classifier *DTC*, the *DSCB* method reaches a certain peak equal to 0.8. At the same point, the other methods do not exceed the value equal to 0.6. Similarly, but in a much smaller scale are the results for the base classifiers *KNN* and *SVM*.

4.2.2 Lessons learned

To conclude the experimental evaluation and analysis, the answers to the research questions posed earlier are presented below:

RQ1: What is the best parameter setup for DSCB?

Selecting the best setup is quite challenging. It should be considered as a more individual task that should be performed before proceeding to the classification, depending on the current problem. However, the tests carried out on the selected pool of data have shown that the proposed method's best settings are to set the parameter γ equal 0.1, balance parameter equal to 0.45, undersampling method as Random Under Sampling and

oversampling method as SMOTE. However, it is worth noting that *DSCB* is a method with very strong adaptability due to its flexible capability of setting various parameters.

RQ2: How robust is the proposed method to data streams with various imbalance ratios?

Experimental evaluation has indicated that *DSCB* is robust to data of varying imbalance ratios. This capability allows the classification of strongly and slightly imbalanced data at a reasonable good level. Even with such a high imbalance ratio where there are only 5% minority class objects in the data stream, the proposed method obtains a very promising result. Such a feature is very desirable, especially among methods designed for data stream classification, where there may be a dynamically changing class imbalance.

RQ3: How resilient is DSCB to label noise on different levels?

Similarly, the robustness of the *DSCB* method to label noise was tested. It is visible that the quality is not so negatively affected. In particular, it is worth noting that initially low noise does not cause much loss for *DSCB*. Only as the noise increases to much higher values, the predictive performance decreases. However, it should be taken into account that data with label noise equal to 40% poses a great challenge for most of the classification models. Despite this high difficulty, the proposed method achieves a quality equal to 0.7 of the $F_1 score$.

RQ4: What is the predictive performance of DSCB for dynamic imbalance ratio data stream?

Subsequent studies have indicated how the proposed method will deal with a stream that has a dynamic data imbalance. Compared to other methods, the *DSCB* achieves high performance and obtains satisfactory scores. Most of the results expressed in different metrics allow the *DSCB* method to have a statistically significant advantage over other methods. Taking into account the results from other experiments leads to the conclusion that the method obtains a higher advantage with dynamically imbalance data streams than with statically imbalanced.

RQ5: What is the proposed method's predictive performance in a comparison to the* state-of-the-art *classifiers?

Comparing the *DSCB* method to the other selected the *state-of-the-art* classifiers one can easily conclude that it performs at a very high level. This refers to both synthetic data and real data. The advantage can also be seen with different types of base classifiers. An important observation is that in the vast majority of the Wilcoxon pairwise rank-sum tests performed on a pool of 450 synthetic data streams, *DSCB* obtains a statistically significant advantage for the $F_1 score$ and $Gmean_s$. It is similar to the *Recall* results, but with a slightly worse overall success rate. It also does not come at a significant cost on the quality expressed by *Precision* and $Specificity$. In the overall evaluation, the method obtains outstanding predictive performance compared to other selected methods of the *state-of-the-art*.

4.3 Deterministic sampling ensemble

Deterministic **S**ampling **E**nsemble (*DSE*) is the chunk based ensemble method for imbalanced data stream classification. *DSE* is the extension of the **D**eterministic **S**ampling **C**lassifier (*DSC*) [28] (Sec. 4.1). *DSC* and *DSE* base on the data accumulation approach [90], that involves collecting minority class data from previous data chunks and using them to balance the current chunk before training a new model. This type of data balancing avoids deleting the majority data that will be used to train the new model as part of the data undersampling process. At the same time, it does not require creating new synthetic objects as it is done during known oversampling methods but uses saved samples from previous data chunks. To obtain much better predictive quality in *DSE* and *DSC*, an accumulation of the whole set of minority class objects is made, and the reduced majority class is also saved. The undersampling method is used to reduce the majority data. This strategy allows us to balance the current data chunk, as well as retain some memory about the distribution of the majority class. We also run some experiments to identify the best oversampling and undersampling methods. This allowed us for the final resignation from the approach based on random oversampling and undersampling, which was used in the *DSC* method. It is worth noting that the proposed method in this article extends the previous idea from a single classifier to the ensemble with majority voting. Another important change that was introduced is a part used to counteract concept drift. Two separate techniques were combined to solve this problem. The first of them is a drift detector to rebuild the entire ensemble and reset the accumulated data when drift is detected. The second is a forgetting mechanism that deletes the oldest models and the oldest accumulated data chunks when the committee reaches a fixed size. The idea of *DSE* is presented in Fig. 4.24, while the Fig. 4.1 and Alg. 5 present its important components.

Proposed method

It is worth paying particular attention to how the drift detection mechanism was designed in this method. There are a few key elements that need to be presented and justified. The most frequently used approach is to observe the base learner's performance and apply appropriate strategies to situations when this performance drops [57]. Similarly, the DSE uses a classifier whose quality is observed and analyzed in each data chunk. Klinkenberg and Renz [133] proposed a drift detection method for sudden concept drift. Their drift detector is based on checking the actual predictive performance and comparing it to the arbitrarily determined value of the threshold. When the quality drops below this value, it means that the concept drift is detected. The threshold is a fixed ratio of the average

quality and quality of the current data chunk. Inspired by this approach, a similar strategy for drift detection was implemented in this work.

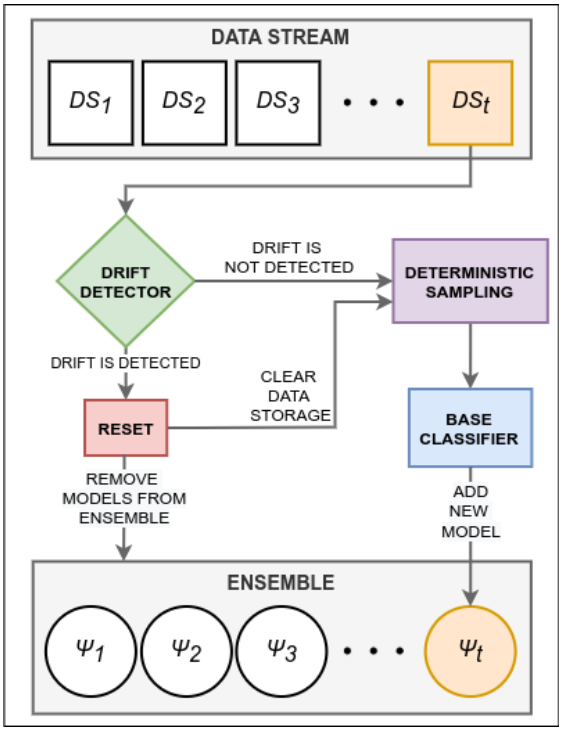


Figure 4.24: *Deterministic Sampling Ensemble*

It will now be described how the drift detector proposed in DSE method works (lines 3 to 13 of the Alg. 5). Each data chunk is sent to the drift detector without processing by the Deterministic Sampling module (Fig. 4.24). In this solution, a classifier in the form of a neural network having one hidden layer with 10 neurons was used. This is a parameter, so it is possible to select a different classifier. When the first data chunk arrives (DS_0), this part is limited only to the drift detection classifier's initialization and training (DDC). Here, any classifier that has the ability to update the model can be used. MLP classifier with one hidden layer and 10 perceptrons was used in the experimental evaluation. However, in every next iteration ($DS_i, i > 1$) when the DDC is trained and can make predictions, its quality is measured. For this purpose, the selected metric is calculated and saved in the $Score$ variable. In prexperiments, the $Gmean_s$ was chosen. Then this value is inserted to the $Score_L$ list. The number of these scores increases with each data chunk. The next step is to calculate the mean of the obtained values and write to $Score_M$. In the Alg. 5 line 7 is the main part of the drift detection mechanism in this method. The $Score$ for the last data chunk is divided by the average values ($Score_M$) from previous data chunks. When this value reaches less than the threshold parameter - T, it means that there has been positive drift detection. In other words, drift is detected when there is a certain drop in quality compared to the average. After drift detection, the associated procedure is started – the memory containing accumulated data

is cleared, all models from the committee are removed, and the DDC prediction quality list from previous data chunks is deleted (Fig. 4.24 – Drift detector). In addition to the drift detector, the oldest models' forgetting was also implemented in this ensemble. A fixed percentage of the quality drop threshold during detection allows good cooperation between two independently performing mechanisms to deal with concept drift. The drift detector has been adapted to react to sudden drifts, where the quality decreases noticeably, and the forgetting mechanism is used for incremental drift.

The rest of the method performs the processing mentioned above and balancing of data chunks. This module is called Deterministic Sampling and it is presented in the detailed diagram in Fig. 4.25. The whole procedure is described in pseudocode lines 14 to 21 of the Alg. 5. The t-th data chunk (DS_t) is balanced using the selected undersampling method (USM) and saved to the t-th undersampled data chunk DSU_t. Any data processing algorithm that allows reducing the size of the majority class can be used as USM. The DSE ensemble classifier is implemented in such a way that the USM method is a parameter that can be easily replaced by any method for the undersampling algorithm during initialization. After undersampling, it is checked whether the accumulated data storage (ADS) already have some stored data. In the case where there are data chunks from previous iterations, the current data chunk is enhanced with whole ADS and overwritten as new DS_t. Then the Imbalance Ratio (IR) of the DS_t is checked. When this value is lower than the Balance Parameter (BP) set during DSE initialization, oversampling is performed using the OSM method. Here, similar to the undersampling, the OSM method is a parameter for the DSE algorithm, and it is selected during initialization. These data are overwritten in place of the current data chunk DS_t. The last step in this part is to save to ADS current data chunk after undersampling DSU_t. This allows storing the entire set of minority class objects and a partial set of majority class objects. The data is now intentionally added to ADS. This avoids learning a new model twice on the same data chunk. If DSU_t was added to ADS immediately after undersampling, the current model would be learned twice using the same data from the minority class and some part of the majority data. It would be possible because the current data chunk is reinforced by the entire data set stored in ADS and the DSU_t undersampled data chunk would already be in ADS. Such a situation is very unwanted and could cause unnecessary confusion during the learning process of the new model.

The final phase of the DSE method is to extend the classifier committee of a new model. This procedure is described in lines 22 to 27 of the Alg. 5. Having already prepared and properly balanced data chunk, it is possible to train the new model. This model is created using a selected base classifier learning method (BCL). Then it is saved to the Ψ_t variable, and added to the ensemble Π. When the size of the classifiers ensemble exceeds the set maximum value s, the oldest model Ψ_{t-s} is removed. A similar operation is

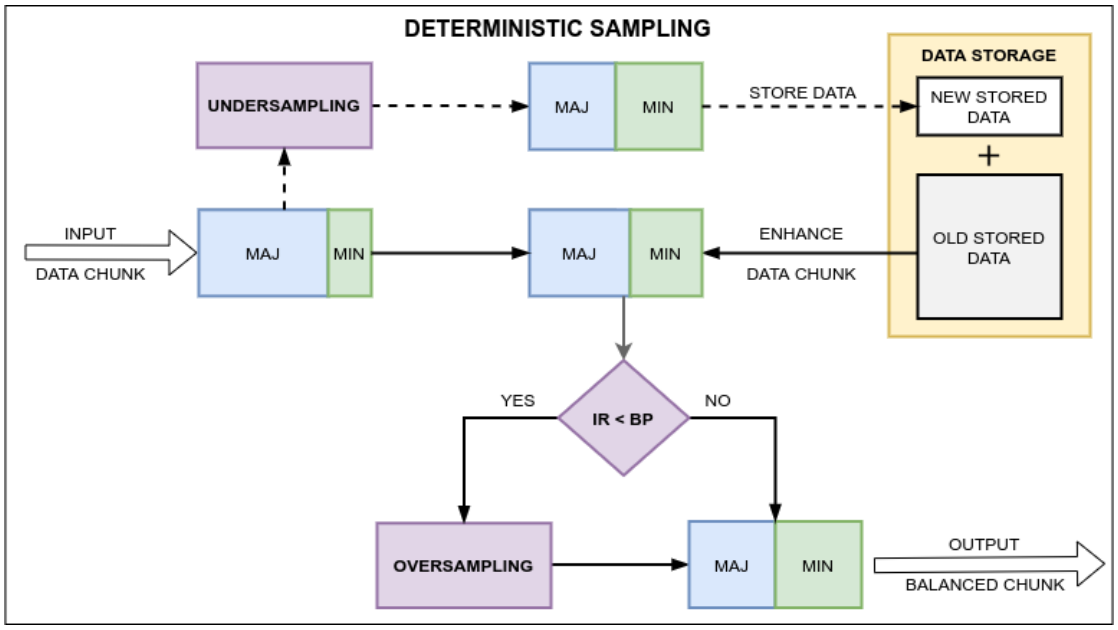


Figure 4.25: *Deterministic Sampling*

performed on ADS where the oldest data chunk DSU_{t-s} is removed to avoid storing data from the entire stream. At the same time, it fulfills a certain function of the forgetfulness mechanism, which counteracts the concept drift problem. The entire process described above is carried out until the entire data stream has been processed.

Computation complexity analysis

Let us present an analysis of the time complexity of the proposed algorithm. The time complexity of this method (T_{DSE}) consists of several components. Time complexity associated with operations on accumulated data storage (T_{ADS}). The time complexity of the drift detector (T_{DDC}). The time complexity of the oversampling method (T_{OSM}). The time complexity of the undersampling method (T_{USM}) and time complexity of the base classifier (T_{CLF}).

$$T_{DSE} = T_{ADS} + T_{DDC} + T_{OSM} + T_{USM} + T_{CLF} \tag{4.4}$$

First of all, it is the complexity associated with accumulated data storage (T_{ADS}). The operations performed on this data are writing and reading. Let us assume that n_s denotes the number of objects in one data chunk. It takes $O(1)$ time to add a new item to the list of any size [54]. When saving new data, the time complexity does not exceed $O(n_s)$, because in the worst case, the whole data chunk will be saved. Reading data from ADS and adding it to the current training data chunk does not exceed the complexity of $O(Sn_s)$, because a maximum S chunks are stored in the memory. The

Algorithm 5 DSE

Input:

DS – data stream
BCL – base classifier learning method
s – maximum size of classifier ensemble
OSM – oversampling method
USM – undersampling method
DDC – drift detector classifier
T – threshold parameter

Symbols:

DS_t – t-th data chunk of data stream DS
ADS – accumulated data storage
DSU_t – undersampled t-th data chunk
$Score$ – drift detector score
$Score_L$ – drift detector scores list
$Score_M$ – drift detector mean score
Ψ_t – t-th model of ensemble

Output:

Π^t – final ensemble for t-th data chunk

for $t = 1, 2, \ldots$ **do**
  if $t > 1$ **then**
    $Score \leftarrow$ Evaluate metric for DDC on DS_t
    $Score_L \leftarrow Score_L \cup Score$
    $Score_M \leftarrow$ Mean $Score_L$
    if $Score/Score_M < T$ **then**
      $ADS \leftarrow \emptyset$
      $\Pi \leftarrow \emptyset$
      $Score_L \leftarrow \emptyset$
    end if
  end if
  Train DDC on DS_t
  $DSU_t \leftarrow$ Undersample DS_t using USM
  if $ADS \neq \emptyset$ **then**
    $DS_t \leftarrow DS_t \cup ADS$
  end if
  if IR of $DS_t < BP$ **then**
    $DS_t \leftarrow$ Oversample DS_t using OSM
  end if
  $ADS \leftarrow ADS \cup DSU_t$
  $\Psi_t \leftarrow$ Train model on DS_t using BCL
  $\Pi \leftarrow \Pi \cup \Psi_t$
  if $|\Pi| > s$ **then**
    $\Pi \leftarrow \Pi \setminus \Psi_{t-s}$
    $ADS \leftarrow ADS \setminus DSU_{t-s}$
  end if
end for

drift detector classifier is another factor affecting the time complexity (T_{DDC}). In this idea, the *Multi Layer Perceptron* classifier with one hidden layer and ten neurons was used. The approximate complexity is $O(n_s p^h)$, where h is the number of hidden layers, and p is the number of neurons in layer. Then these are the data sampling methods (T_{OSM}, T_{USM}). The complexity affecting the entire method is highly dependent on their selection. After carrying out the experiments for selecting the best parameters, the SVMSMOTE method for oversampling was selected. The complexity of this method is $O((n^m(R+1)+n^M)^3)$, where n^m is the number of minority class objects and n^M is the number of majority class objects, and R is the fraction of additional randomly selected minority class samples [244]. Assuming that the minority class does not count more than 50% of all samples, in the worst case, this method has the complexity of the size $O(1\frac{1}{2}n_s^3)$. The Neighborhood Cleaning Rule method was selected for undersampling. The complexity is $O(n_s^2)$ because this method must find the nearest neighbors for each point in the data chunk. The last element affecting the final method complexity is the base classifier (T_{CLF}). Four different classifiers were used during the experiments – *Support Vector Machine* (SVM), *k-Nearest Neighbors* (KNN), *Gaussian Naïve Bayes* and *Decision Tree CART*. *SVM* has a complexity of $O(n_s^3)$ [1]. The complexity of the KNN Classifier and the *Gaussian Naïve Bayes* is equal to $O(n_s d)$, where d is the number of dimensions [59]. The complexity of the CART is $O(dn_s^2)$ [238]. After substituting the individual components of the time complexity of *DSE* method into the formula, the result is:

$$T_{DSE} = O(10n_s) + O(10n_s) + O(1\frac{1}{2}n_s^3) + O(n_s^2) + O(n_s^3) \tag{4.5}$$

The configuration on which the experiments were carried out in this article has a complexity of $O(1\frac{1}{2}n_s^3)$, because it is the largest complexity of all components. It should be noted that the method has been designed in a flexible way. It allows the selection of other base classifiers and oversampling or undersampling methods, depending on the needs of lower time complexity.

4.3.1 Experimental evaluation

The experiments carried out in this article were mainly intended to check how the proposed method for classifying imbalanced data streams works compared to other *state-of-the-art* methods. The following research hypotheses were formulated:

RQ1: What is the best setting of DSE, i.e., how proposed method works with different base classifiers and different preprocessing methods?

RQ2: How flexible is DSE to changing the imbalance ratio?

RQ3: How robust is proposed method to label noise?

RQ4: What is the predictive performance of the proposed method in a comparison to the *state-of-the-art* classifiers?

Setup

The experimental environment has been implemented in Python programming language and is publicly available on the Github repository[3]. The research was carried out using four basic classifiers. Used implementations are from scikit-learn [200] machine learning library for the Python. Each tested ensemble method built a maximum of 10 models of the base classifier during the experiment on one data stream. List of selected base classifiers:

- *k-Nearest Neighbors* (KNN)
- *Support Vector Machine* (SVM)
- *Gaussian Naive Bayes* (GNB)
- *Decision Tree CART* (DTC)

Evaluation was executed in the *test-then-train* manner [24]. Each data chunk except the first is used for testing the model and then for training. This approach allows the entire data stream to be used without splitting the data into training and test sets. The experiments were evaluated on the basis of five different metrics – $Gmean_s$ [152], F_1score [235], $Recall$, $Specificity$, $Precision$. The proposed method has been tested and compared with the performance of selected *state-of-the-art* methods for the classification of imbalanced data streams. To obtain the possibility of adequate comparison, all comparative experiments were also performed with these methods. Besides, the multilayer neural network classifier was used as the baseline. Below is a list and acronyms for these methods used later in this work:

[3]Repository link: https://github.com/w4k2/DSE

- *REA* – Recursive ensemble approach [51]
- *KMC* – K-means clustering undersampling ensemble [264]
- *L++CDS* – Learn++CDS [62]
- *L++NIE* – Learn++NIE [62]
- *OUSE* – Over and under-sampling ensemble [89]
- *MLPC* – Multi-layer perceptron classifier

The main purpose of the experimental evaluation is to check the predictive performance of the proposed method and compare it with *state-of-the-art* methods. For this purpose, extensive research has been conducted on generated and real data streams (Tab. 3.2). The size of the data chunks in real data was chosen experimentally. Synthetic data streams come from two different generators – MOA [24] and stream-learn [150]. All data streams are two-class problems. Separate sets of data streams have been prepared for each experiment which differ in parameter setup. Tab. 4.1 presents parameters such as a number of objects, chunk size, imbalance ratio, noise level or drift type of generated synthetic data streams.

Parameter	Value		
Number of samples	100000		
Number of chunks	200		
Chunk size	500		
Number of classes	2		
Number of features	10 (8 informative + 2 redundant)		
Number of drifts	1	5	
Concept drift types	sudden	incremental	
Random state	1111	2222	
Label noise	0%	10%	
Imbalance ratio	10%	20%	30%

(a) *stream-learn*

Parameter	Value		
Generator type	RandomRBFGenerator		
Number of samples	100000		
Number of chunks	200		
Chunk size	500		
Number of classes	2		
Number of features	10 (8 informative + 2 redundant)		
Number of drifts	1	5	
Concept drift types	sudden	incremental	
Random state	1111	2222	
Label noise	0%	10%	
Imbalance ratio	10%	20%	30%

(b) *MOA*

Table 4.1: *Parameter setup for data stream generators*

Experiment 1 - Parameter setup

The first experiment was carried out to determine the best hyperparameters of the proposed method. This will allow us to choose the right settings so that the method is presented from the best side. The plan of this experiment is to set hyperparameters

globally, so-called best default settings. The method has 5 different hyperparameters. The maximum number of models and the base classifier are depending on the assumptions of the experiment and the size of the data, so they will not be included in this study. The next is the level of imbalance ratio of oversampling to increase balance. The last two are undersampling and oversampling algorithms. Four different data streams were used for this experiment. Two different types of drift – incremental and sudden and two different generators MOA and stream-learn. All streams in this part had 10000 samples, 10 attributes, 10% imbalance ratio and 0% label noise. The results for the combination of selected oversampling and undersampling methods [79] were evaluated and the results analyzed. The medians were calculated from the obtained metrics values for each data stream separately. Then the average for all streams with one base classifier was calculated. The tables show the results obtained from this experiment grouped by base classifier and metric. List of selected undersampling methods:

- Condensed Nearest Neighbors (CNN)
- Edited Nearest Neighbors (ENN)
- Repeated Edited Nearest Neighbors (RENN)
- All KNN (AllKNN)
- Instance Hardness Threshold (IHT)
- Near Miss (NM)
- Neighbourhood Cleaning Rule (NCR)
- One Sided Selection (OSS)
- Random Under Sampler (RUS)
- Tomek Links (TL)

List of selected oversampling methods:

- ADASYN (ADASYN)
- Borderline SMOTE (BSMOTE)
- Random Over Sampler (ROS)
- SMOTE (SMOTE)
- SVMSMOTE (SVMSMOTE)

The first analysis of the results focuses on the selection of the best oversampling method. At this point, Random Undesampling is treated as the starting solution. At first glance, it is hard to pick the best candidate (Fig. 4.26). The differences between the methods are quite subtle. After a more in-depth analysis, SVMSMOTE obtained the highest quality for different metrics. Therefore, this method was chosen for further research.

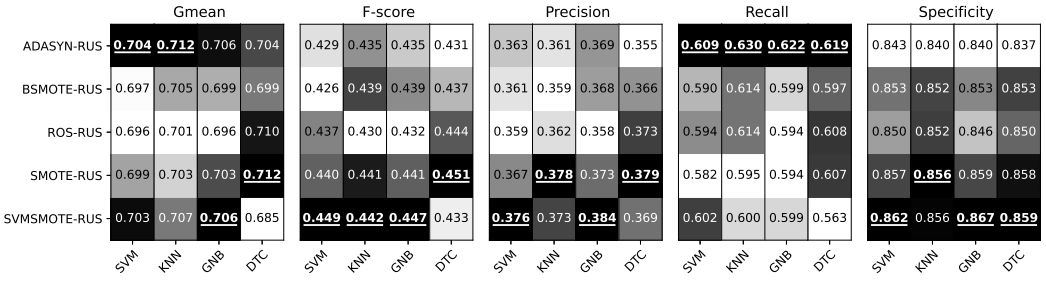


Figure 4.26: *Results of Random Under Sampling combination with oversampling methods. Darker is better, best value is bold and underscored*

In the next step, the best method for data undersampling was determined. This experiment focused on finding the best result for the combination of the SVMSMOTE method with the selected undersampling method. For this purpose, all possible variations were tested. At first glance, it can be seen that the results do not differ significantly (Fig. 4.27). Depending on the chosen metric, different method achieves the best results in this test. However, it can be seen that for $Gmean_s$ and *Recall* the *Neighborhood Cleaning Rule* (NCR) method has the best result and rest of the metrics do not stand out significantly from the other methods. The choice of NCR will complement the overall quality classification with SVMSMOTE, which has the opposite tendency to achieve good results in *Precision* and *Specificity*. Then all possible combinations of the NCR method with other oversampling methods were compared. This test confirmed the previous choice that the NCR method is best suited for SVMSMOTE (Fig. 4.28).

The last of the hyperparameters is Balance Parameter (BP). It is a parameter determining the threshold of data imbalance, when the *DSE* method has to additionally balance the data chunk with oversampling. The BP threshold is compared to the imbalance ratio of the data chunk after strengthening it using samples stored in ADS. Unfortunately, the obtained results do not allow to indicate the best value (Fig. 4.29). Depending on the analyzed metric, the BP hyperparameter goes into high or low values for the best result. Finally, a decision was made to select and maintain the value initially adopted – 50% objects of each class. Because the tests carried out do not allow for the explicit selection of the best value, the method will aim for a perfect class balance.

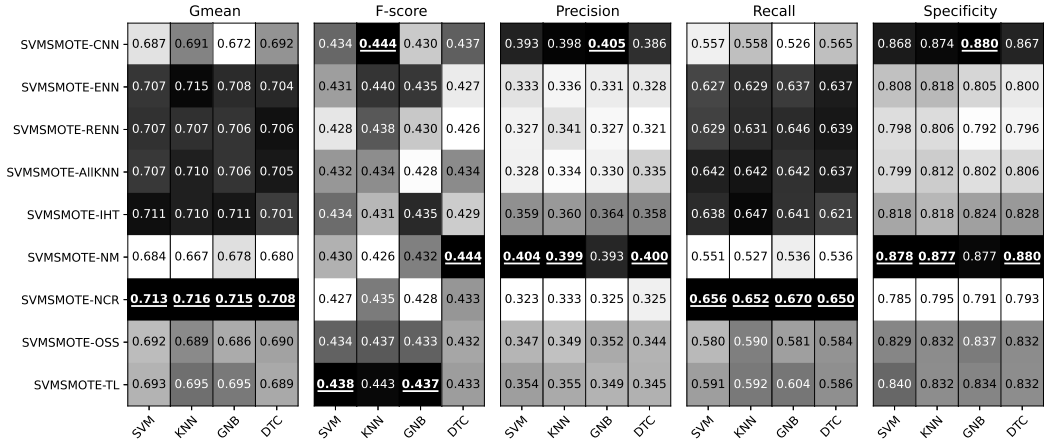


Figure 4.27: *Results of SVMSMOTE combination with undersampling methods. Darker is better, best value is bold and underscored*

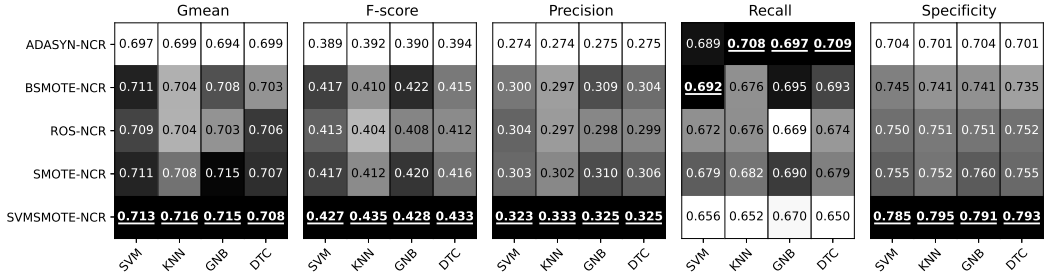


Figure 4.28: *Results of NCR combination with oversampling methods. Darker is better, best value is bold and underscored*

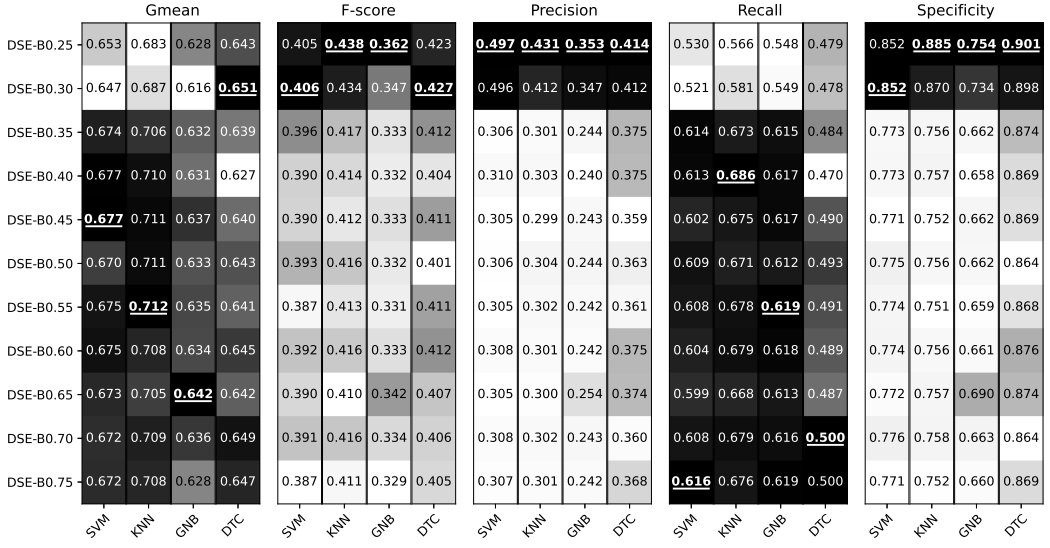
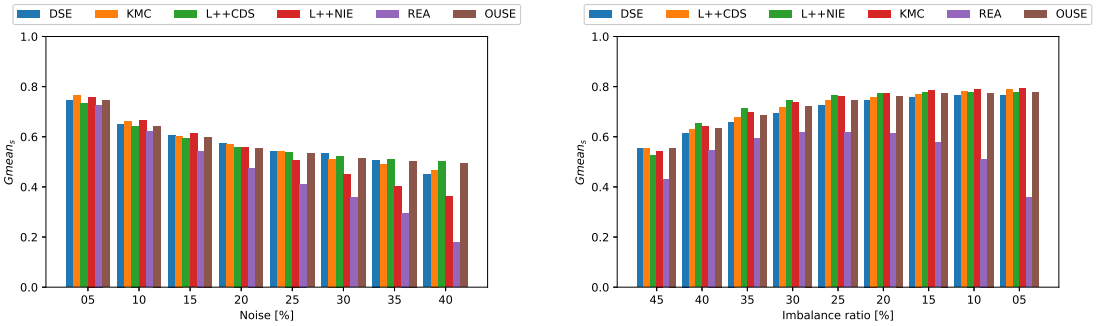


Figure 4.29: *Balance parameter setup experiment. Darker is better, best value bold and underscore*

Experiment 2 - Label noise and imbalance ratio

In this experiment, the effect of changing the imbalance ratio and label noise of a data stream was examined. For more accurate results without major interference, the tested



(a) Label noise *(b) Imbalance ratio*

Figure 4.30: *Selected results from noise and balance experiments (Gmean$_s$ metric and KNN base classifier)*

data did not have any concept drift. Imbalance ratio and label noise varied depending on the tested property. The results obtained were averaged over the entire stream.

The imbalance ratio of the data stream has an impact on the quality of the proposed method. In the presented plot (Fig. 4.30b) it can be seen that with only 5% of minority class objects the quality in comparison to an equally balanced stream significantly decreases. Similarly, the phenomenon can be observed in the case of other metrics or other base classifiers used in this experiment. However, an essential observation is that this problem also applies to compare other methods. It means that the data difficulty associated with a strong imbalance does not cause a very negative effect in the *DSE* algorithm, and compared to other methods, remains at a similar level. Looking at the other methods, REA behaves quite atypically, and its quality decreases at a much lower data imbalance. This may indicate that the method was designed to operate on strongly imbalance data. From the whole pool of methods, the best results in this experiment were obtained by KMC.

A similar situation occurs in research with various label noise (Fig. 4.30a). The experiment was carried out to check how much the level of noise affects the quality of methods. The range of the tested area is from 5% to 40% of the label noise in the stream. Higher noise seems unnecessary because the stream begins to become very random. It follows that the higher the noise, the lower the results obtained. This applies to all tested scenarios with various base classifiers and metrics. In the presented results, the most vulnerable methods are Learn++NIE and REA. OUSE and Learn++CDS show the highest noise immunity. The *DSE* and KMC is roughly in the middle.

Experiment 3 - Synthetic data

In this experiment full set of data will be used (Tab. 4.1). Evaluation of all streams will generate substantial number of results. This must be multiplied by four variants of the base classifier and the calculation of five different metrics. This makes it impossible to review all results one by one. Aggregation of metrics into averages will not allow us to draw accurate conclusions. For the analysis of such results, a special statistical tool should be used, which will collect everything on several charts and will not cause a significant loss of information. The main intention of this experiment is to compare *DSE* with other methods. For this purpose, the results obtained were analyzed using the Wilcoxon rank-sum pair statistical tests divided into different streams. Then the obtained results will be collected into win-tie-loss charts, which will allow an easy way to assess the quality of the tested method in a given metric compared to other methods. Additionally, critical difference values were determined for each chart, indicating whether the compared *DSE* method is statistically better with the results obtained in this experiment.

In the beginning, the results of synthetic streams with the incremental drift of the concept are analysed (Fig. 4.31). Looking at the graph, it can be seen that for incremental drifts *DSE* achieves slightly weaker results when it uses Gaussian Naive Bayes as the base classifier. The second observation is that with some methods it loses in $Specificity$ and $Precision$. In contrast, *DSE* in $Gmean_s$, F_1score and $Recall$ for almost every base classifier is statistically better in these data streams.

The next picture shows the results obtained for the generated sudden data drift streams (Fig. 4.32). It is evident that compared to the incremental drift, the proposed method is doing a little worse, but it is not much worse. Still, in comparison to other methods for $Gmean_s$, F_1score, and $Recall$, in most cases, *DSE* is significantly statically better for the results obtained in this experiment. However, $Precision$ and $Specificity$ are at a lower level compared to other methods. It can be seen that Learn++CDS is one of the best methods in this experiment. These results show a very interesting observation about the REA method. The method seems to be much better than *DSE* in $Recall$. The situation is entirely opposite for $Precision$ and $Specificity$, where REA has very low results. This method has a very high tendency to classify objects as a minority class. This is an example of a method that tries to balance the data excessively, which results in a bias toward the minority class. This will result in a poor predictive quality.

It is worth analyzing and presenting an example graphs (Fig. 4.33) of a selected metric over the data chunks of one data stream. A data stream with five concept drifts was selected. The plot and results were previously smoothed using a Gaussian filter for better readability. The graph shows that the *DSE* method is performing very well. In the drift

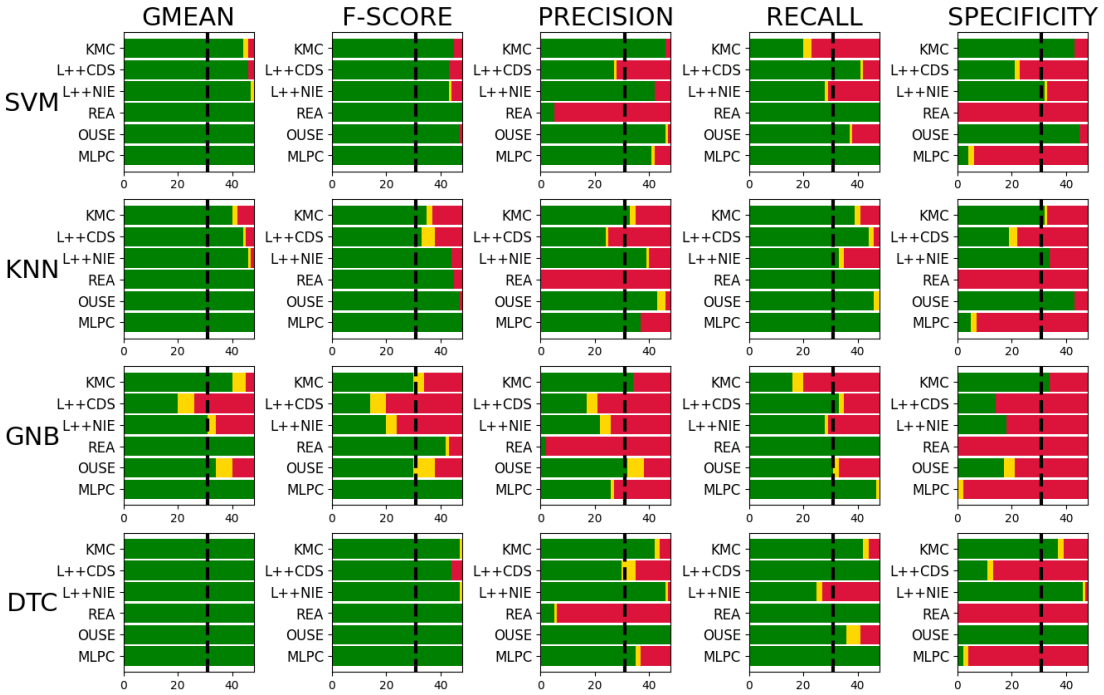


Figure 4.31: *Wilcoxon pair rank-sum tests for synthetic data streams with incremental concept drift. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*

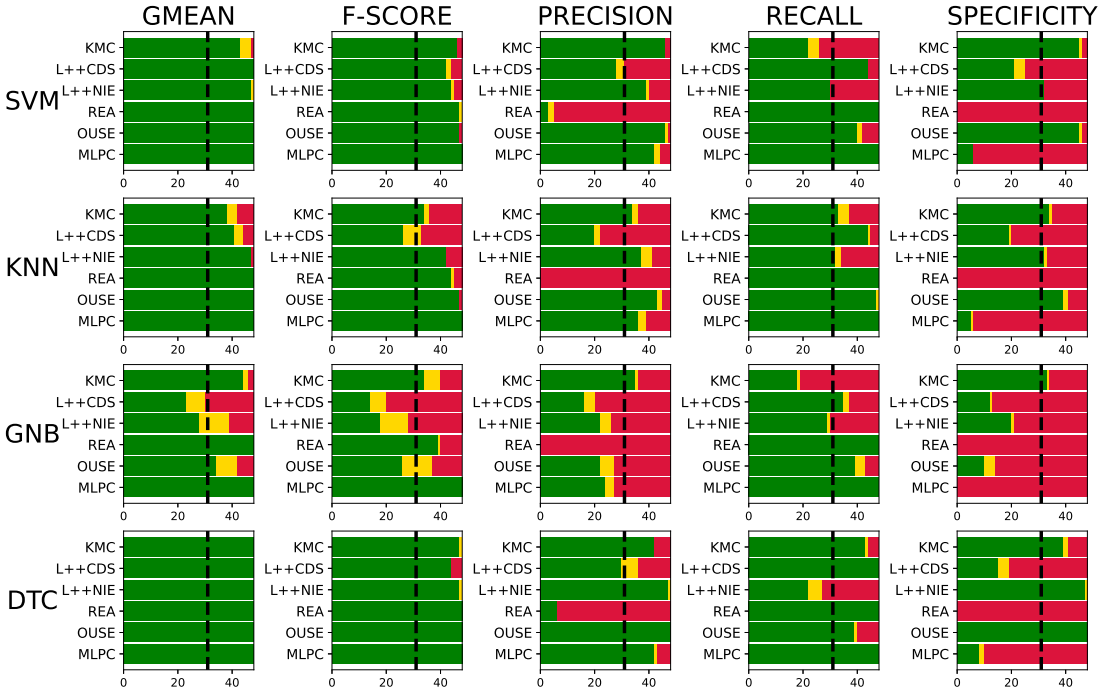
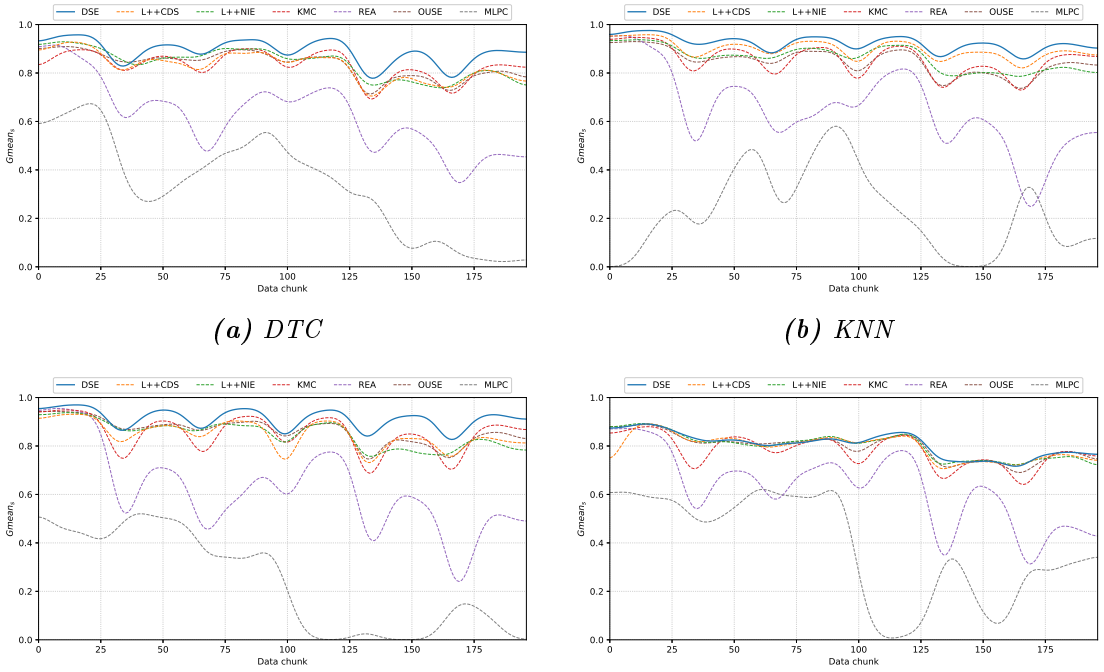


Figure 4.32: *Wilcoxon pair rank-sum tests for synthetic data streams with sudden concept drift. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*



(a) DTC *(b) KNN*

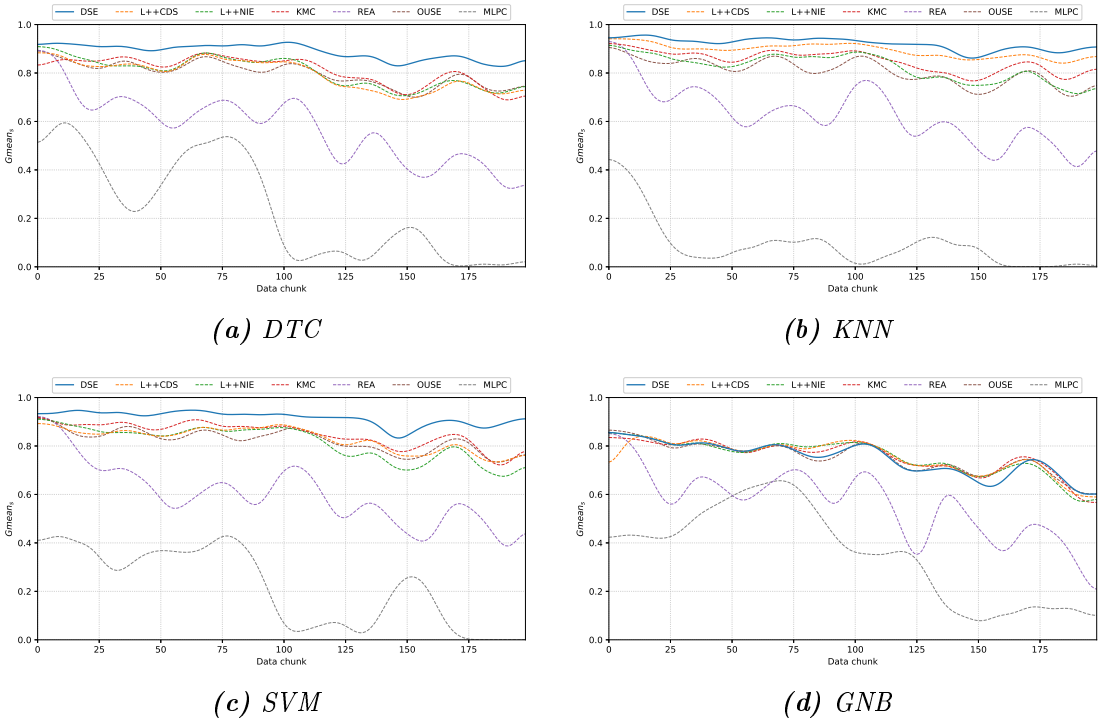
(c) SVM *(d) GNB*

Figure 4.33: *$Gmean_s$ score over the data chunks for synthetic data with sudden drift*

spots there are quality drops. It is understood in case of sudden drift. However, it can be seen that *DSE* most often has the smallest drop compared to other methods. This means good adaptability of the method to sudden drift. It is also visible that REA performs the worst, not taking into account *MLPC*, which is the baseline classifier. Other methods are at a fairly similar level. Slightly larger drops after drift can be observed among the KMC and Learn++CDS methods compared to Learn++NIE and OUSE, which perform more stable after drift. Similar results are achieved for incremental data stream (Fig. 4.34).

Additionally, tests were carried out on the metrics presented in Sec. 2.7 to analyze behavior of the method with concept drifts. We used the maximum performance loss MPL (Eq. 2.23) and restoration time RT (Eq. 2.22). Both metrics are calculated based on $Gmean_s$. The results are presented in Tab. 4.2,– 4.5.

Most of the results obtained are similar. It is noticeable that the Tab. 4.2 presenting the results of performance loss on generated incremental streams shows some advantage of the *DSE* method over the others. The other results do not show a clear leader, but most often, the L++CDS method dominates. It is worth noting that the presented average results do not differ much from each other, and the standard deviation is high compared to the mean values.



(a) DTC *(b) KNN*

(c) SVM *(d) GNB*

Figure 4.34: *$Gmean_s$ score over the data chunks for synthetic data with incremental drift*

Table 4.2: *Mean performance loss with standard deviation on generated incremental streams (less is better, best result in bold)*

	SVM	KNN	GNB	DTC
DSE	**0.0677±0.0390**	**0.0462±0.0209**	0.0744±0.0304	**0.0598±0.0336**
KMC	0.0867±0.0823	0.0542±0.0219	**0.0658±0.0225**	0.0705±0.0279
L++CDS	0.0730±0.0290	0.0523±0.0214	0.0821±0.0403	0.0835±0.0511
L++NIE	0.3670±0.3862	0.0835±0.0634	0.1319±0.1147	0.0785±0.0441
REA	0.4233±0.2916	0.2033±0.1029	0.3675±0.2270	0.3542±0.2639
OUSE	0.1205±0.0552	0.0978±0.0338	0.0930±0.0332	0.0908±0.0261
MLPC	0.5916±0.4081	0.5737±0.4234	0.5970±0.3886	0.5747±0.3803

Experiment 4 - Real data

The last experiment focus on the real data streams (Fig. 4.35). The difference that is noticeable compared to the other results is quite a lot of draws. This may be due to the small size of these data streams, which caused similar method performance. However, in most combinations, the *DSE* method achieves statistically better or the same predictive quality compared to the results of other methods. It means that *DSE* performs on real data streams just as well as the *state-of-the-art* methods. For these data streams, there

Table 4.3: *Mean performance loss with standard deviation on generated sudden streams (less is better, best result in bold)*

	SVM	KNN	GNB	DTC
DSE	0.2508±0.2070	0.2086±0.1464	0.1908±0.1323	0.2213±0.1391
KMC	0.2366±0.2076	0.2090±0.1520	0.1843±0.1581	0.1986±0.1137
L++CDS	0.2486±0.2257	0.1833±0.1068	**0.1832±0.1185**	0.2072±0.0913
L++NIE	0.4574±0.3171	**0.1814±0.0797**	0.2012±0.0963	**0.1740±0.0620**
REA	0.5711±0.2896	0.3426±0.1948	0.4495±0.2125	0.4144±0.2003
OUSE	**0.2278±0.1250**	0.2166±0.1415	0.1862±0.1115	0.1771±0.0597
MLPC	0.6394±0.3737	0.6052±0.3624	0.5881±0.3890	0.6085±0.3738

Table 4.4: *Mean recovery time with standard deviation on generated incremental streams (less is better, best result in bold)*

	SVM	KNN	GNB	DTC
DSE	0.0078±0.0047	0.0060±0.0018	0.0069±0.0035	0.0062±0.0025
KMC	0.0091±0.0054	0.0067±0.0031	0.0063±0.0026	0.0066±0.0030
L++CDS	0.0072±0.0030	**0.0055±0.0010**	0.0064±0.0024	**0.0062±0.0018**
L++NIE	0.0074±0.0036	0.0073±0.0035	0.0071±0.0033	0.0070±0.0028
REA	0.0083±0.0041	0.0079±0.0038	0.0083±0.0035	0.0087±0.0042
OUSE	**0.0064±0.0024**	0.0076±0.0034	**0.0063±0.0021**	0.0069±0.0032
MLPC	0.0112±0.0076	0.0102±0.0063	0.0099±0.0056	0.0102±0.0067

Table 4.5: *Mean recovery time with standard deviation on generated sudden streams (less is better, best result in bold)*

	SVM	KNN	GNB	DTC
DSE	0.0082±0.0035	0.0067±0.0028	0.0085±0.0042	0.0093±0.0050
KMC	0.0123±0.0070	0.0100±0.0047	0.0095±0.0043	0.0115±0.0064
L++CDS	**0.0066±0.0018**	**0.0058±0.0012**	**0.0064±0.0016**	0.0071±0.0018
L++NIE	0.0067±0.0025	0.0059±0.0013	0.0065±0.0016	**0.0064±0.0017**
REA	0.0190±0.0148	0.0179±0.0135	0.0152±0.0098	0.0157±0.0108
OUSE	0.0075±0.0022	0.0088±0.0039	0.0072±0.0017	0.0067±0.0016
MLPC	0.0141±0.0111	0.0154±0.0121	0.0160±0.0128	0.0132±0.0091

is no apparent weakness in the results of $Precision$ and $Specificity$, which were visible in the synthetic data streams.

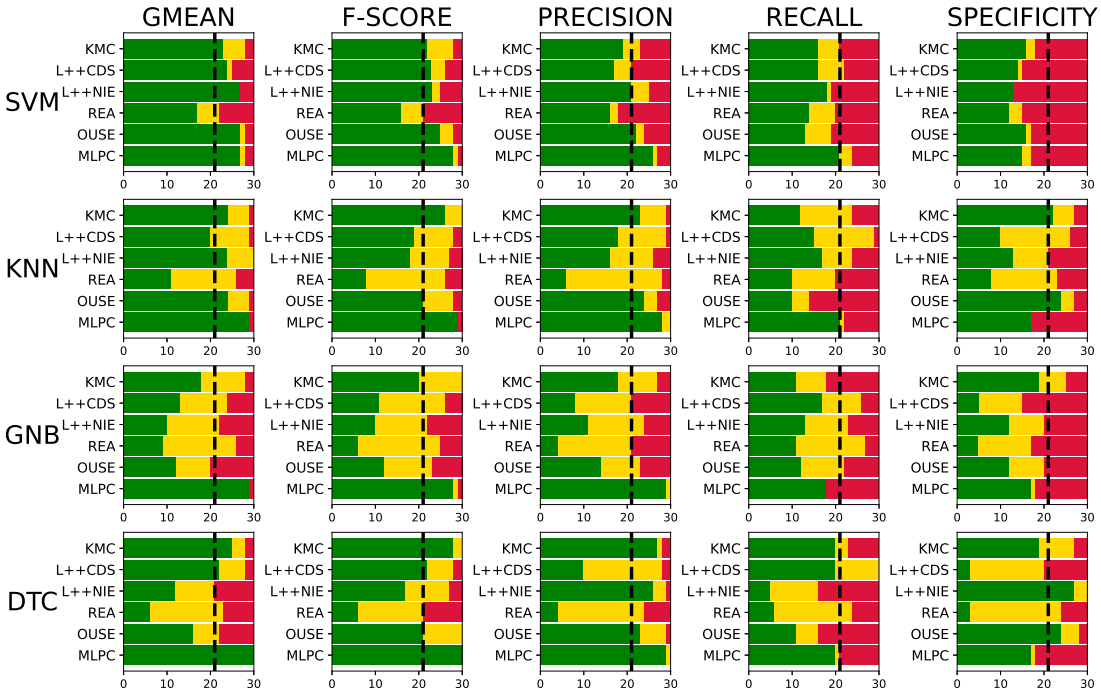


Figure 4.35: *Wilcoxon pair rank-sum tests for real data streams. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*

4.3.2 Lessons learned

To summarize the experimental evaluation conducted, the answers to the research questions posed earlier are presented below:

RQ1: What is the best set of DSE, i.e., how proposed method works with different base classifiers and different preprocessing methods?

The *DSE* method is strongly parameterized. It required conducting appropriate experiments to identify the best hyperparameters. The result is the choice of Neighborhood Cleaning Rule as the undersampling method and SVMSMOTE as the oversampling method. Unfortunately, the best setup for Balance Parameter (BP) has not been determined in analysis. However, 50% of the objects in each class were selected as the default value. The justification for this choice is to aim for an ideal class distribution.

RQ2: How flexible is DSE to changing the imbalance ratio?

Tests carried out indicate that a change in the imbalance level does not have a very negative impact than other tested methods. There is a certain decrease in the quality of the prediction, while the level of imbalance increases. It is a fairly obvious phenomenon

because the problem's difficulty increases significantly, which is reflected in a decrease in quality.

RQ3: How robust is proposed method to label noise?

The *DSE* is also very robust to the label noise. Similar to imbalance ratio, the method reacts to higher label noise with a slightly lower quality of prediction. When the noise increases, there is a decrease in predictive performance. However, *DSE* does not differ much from the others. High label noise has a negative impact because it also increases the difficulty of the problem.

RQ4: What is the predictive performance of the proposed method in comparison to the state-of-the-art classifiers?

Based on the conducted experiments, one can observe that the proposed method works very well for both noisy and no-noise data. The obtained results show that the *DSE* method is statistically better than the selected *state-of-the-art* methods for most data streams on which it was tested using various base classifiers, especially using Decision Tree CART and Support Vector Machine. *DSE* achieves good results with both sudden and incremental drifts. Looking through the selected metrics' prism, the proposed method performs the least well for a $Specificity$ score compared to other methods. However, the high level of $Gmean_s$, F_1score, and *Recall* indicates that this is fairly well compensated for the correct minority class prediction. It is quite an important feature of the method for imbalanced data because the cost of making a mistake is much higher for objects of a minority class than for a majority class. The tests performed on the performance loss and recovery time did not indicate the best performing method clearly, but *DSE* is not doing the worst.

4.4 Deterministic sampling ensemble of one class support vector machine classifiers

The proposed method is designed to classify the imbalanced data streams. This data is associated with many difficulties which this method will attempt to solve. The main idea is to combine the best techniques derived from two solutions that have also been designed for imbalanced data stream classification. These approaches, despite having many features in common, are significantly different. The first one is **D**eterministic **S**ampling **E**nsemble – *DSE* (Sec. 4.3), a method that focuses on using oversampling and undersampling of data along with the accumulation of data from previous chunks, to use this knowledge in further stages of learning. The second **O**ne **C**lass support vector machine classifier **E**nsemble for **I**mbalanced data **S**tream – *OCEIS* (Sec. 3.1) approach, which uses One Class *SVM* models built on different feature subspaces determined by clustering. To better understand the idea of combining these two methods and to point out the novelties introduced to the method, the whole concept will be presented with more details.

The **D**eterministic **S**ampling **E**nsemble of **O**ne **C**lass Support Vector Machine classifiers (*DSE-OC*) is a method which processes the data stream in a chunk-based manner (Fig. 4.36 a). The main foundation of it is to use a data accumulation approach from previous data chunks [90]. These saved data are then used to increase the number of minority class objects in subsequent chunks (Fig. 4.36 b). This can be called an semisynthetic oversampling technique. It has many benefits when looking at the data distribution, which is affected by the imbalance. First, the issue of uneven class distribution is solved. For this, oversampling is used, but without creating new artificial samples. Minority data is amplified using older minority data from earlier iterations of the data stream. This gives quite a lot of potential to this method because no new patterns are produced that may not necessarily fit into the current distribution. At the same time, there is no need to apply undersampling methods to the current data chunk, which could involve the loss of information.

This idea is not a complete novelty, but most of the authors using this strategy decided to accumulate data only from the minority class [51]. The innovation is to accumulate data from the majority class, but in reduced quantities. This reduction in number is done by the undersampling method, where it aligns with the number of minority class objects. It is also worth noting that extending data accumulation to selected subsets of the minority class allows to ensure continuity with the stored data. Saving only samples from one class could lead to significant decomposition disturbances. This could lead to

a kind of imbalance in the other direction, where after introducing too many patterns from the minority class, it would dominate the majority class.

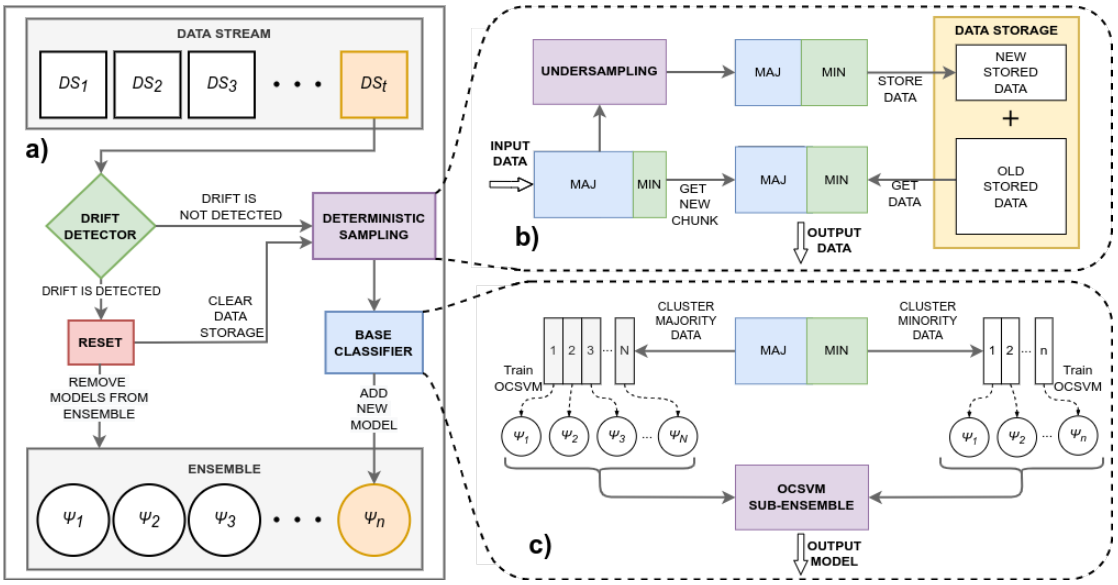


Figure 4.36: *Overview diagram of* DSE-OC *method*

The second crucial core of this method is an modification of the OCEIS method used as a base classifier (Fig. 4.36 c). The approach focuses on the use of one-class *SVM* (OCSVM) models for solving the problem of classifying a binary imbalanced data stream. Despite the many challenges posed by this difficult data, the method also faces the issue of using *OCSVM* on binary data. Originally, the *OCSVM* method was designed to solve anomaly detection [75] or other related to this task. However, one can see some new trends of applying such models to classify binary data [131] [277]. Such application requires appropriate steps to decompose the data and then combine the models into one decision.

To obtain the predictive ability of one-class models in binary problems, it is necessary to split the training set into two different classes. Then each model is learned on one set and then a decision is made based on what both classifiers indicate. This idea is good, until at the same time both classifiers indicate membership in their decision area or worse both indicate no membership. Then there is the serious problem of a sample that has not been clearly classified into either of the two possible options and remains unlabeled. The solution to this may be the use of supports that are returned by one-class models. Then, with a certain margin of error, it is possible to determine the higher or lower probability of an object belonging to a given model. Unfortunately, most of this type of models have some limitations that do not allow us to accurately indicate the probability of belonging due to the lack of counterexamples. The similar happens in this

method. However, instead of using the support function, some measure of the distance to the decision boundary is used, which is defined by Eq. 2.10.

An essential condition that must be ensured when designing the ensemble method is adequate model diversification. One of the approaches that will allow this is to build models on subsets of data. Such subspaces can be obtained by clustering. However, there is a problem of determining the number of clusters that should be created. The original approach implies that the number of clusters should be selected before applying the algorithm to data learning. The most preferable way is to automatically select the best number of clusters. The main motivation is that data describing various problems have very different characteristics of data distribution and class sizes. This problem is common in a data stream where the distribution can change in time. At this point, finding the best number of clusters is an issue. If the majority data will be divided into more clusters, it will be a sort of data balancing by segmentation. On the other hand, when the minority data will be divided into the same number of subsets, the problem returns to the original imbalance ratio or even worsens in some cases.

This raises the idea that it is best to determine these numbers for each class separately. However, doing this manually would require much work and finally it would not be a good solution from the perspective of using this method for further development. Hence, an idea emerged to do some automation, which would allow, in a dynamic way, to make a fast and relatively successful optimization of the number of clusters for each set of patterns from a given class. Such choices are made possible by using a metric that provides some measurable indicator of the clusters' quality. For example, the Silhouette Coefficient [165] metric can be used (Eq. 3.1 in Sec. 3.1).

Procedure

For better understanding, the entire algorithm procedure will be presented descriptively along with psuedocode (Alg. 6). In the first iteration, the part responsible for drift detection is omitted. Therefore, it can be said that the core of the algorithm starts from line 13 of the pseudocode. In the first step, the classifier responsible for drift detection (DDC) is trained. The whole algorithm is designed in such a way that this is treated as a certain parameter with the possibility to change it easily. However, it must be a classifier with the ability to be trained. In this implementation proposed here, an MLP classifier with one hidden layer and ten perceptrons was used. In the next step, the data chunk is processed using the undersampling method (USM) in such a way that the equal imbalance ratio of the data is achieved and saved in DSU_t. However, it is important to note that the original data chunk is maintained. Then, when the accumulated data

storage (ADS) is not empty, this original data chunk is amplified with the samples from there. It is obvious that in the first iteration this will not be possible. After this step is the undersampled data chunk (DSU_t) is added to the accumulated data storage (ADS). This prevents the original data chunk from being enhanced with its undersampled version.

With the data already prepared, *DSE-OC* proceeds to build the model using the base classifier. This whole process is described by Alg. 7. First, the data is divided into two sets of samples from the minority class and the majority class. Then, these partitioned data are clustered using the K-Mean algorithm. However, this method requires an input of the number of clusters into which the submitted data set will be divided. This number is determined in a dynamic way. Minority data and majority data are separately segmented into clusters whose number varies from 2 to K. Values are then computed for the created sets using a selected metric that determines the consistency of the clusters. Those with the best metric scores are passed on as the solution to select the optimal number of clusters, respectively, for each class. In some ways, this is a brute-force check of all possibilities with a maximum bound equal to K. However, it allows to determine the number of clusters in a dynamic way. *OCSVM* models are then built on these data clusters. In this way, a subensemble consisting of one class classifiers is generated.

Then, once a model of the base classifier is built, which is a kind of subensemble, the algorithm adds it to the main ensemble. In the last step, it is checked if the size of this main ensemble does not exceed a given value. When it does, the oldest models are removed. At the same time, the oldest chunks are removed from the accumulated data storage to avoid storing data from the entire stream, which is a very undesirable practice.

Moreover, the base classifier is itself an ensemble of classifiers, which requires the design of a decision rule to determine the final decision. For such an ensemble, the problem is much more complicated, because it consists of one-class models that are intended to perform the prediction on the binary data. The approach based on majority voting or support accumulation will not produce good results. The *OCEIS* uses the original combination rule based on distance from the decision function which determines the relative distance of classifiers boundries to predicted samples. Let f_i^m denotes decision function of classifier Ψ_i^m (see Eq. 2.10) and f_j^M of classifier Ψ_j^M respectively. The final decision of *OCEIS* is made according to the following classification rule:

$$\Psi(x) = \begin{cases} \text{minority class} & if \max\limits_{i \in \Pi^m} \left(f_i^m(x)\right) \geq \max\limits_{j \in \Pi^M} \left(f_j^M(x)\right) \\ \text{majority class} & \text{otherwise} \end{cases} \tag{4.6}$$

Algorithm 6 DSE-OC

Input:

DS – data stream
DS_t – t-th data chunk of data stream DS
s – maximum size of classifier ensemble
OSM – oversampling method
USM – undersampling method
DDC – drift detector classifier
T – threshold parameter

Symbols:

ADS – accumulated data storage
DSU_t – undersampled t-th data chunk
$Score$ – drift detector score
$Score_L$ – drift detector scores list
$Score_M$ – drift detector mean score
Ψ_t – t-th model of ensemble

Output:

Π^t – final ensemble for t-th data chunk

for $t = 1, 2, \ldots$ **do**
  if $t > 1$ **then**
    $Score \leftarrow$ Evaluate metric for DDC on DS_t
    $Score_L \leftarrow Score_L \cup Score$
    $Score_M \leftarrow$ Mean $Score_L$
    if $Score/Score_M < T$ **then**
      $ADS \leftarrow \emptyset$
      $\Pi \leftarrow \emptyset$
      $Score_L \leftarrow \emptyset$
    end if
  end if
  Train DDC on DS_t
  $DSU_t \leftarrow$ Undersample DS_t using USM
  if $ADS \neq \emptyset$ **then**
    $DS_t \leftarrow DS_t \cup ADS$
  end if
  $ADS \leftarrow ADS \cup DSU_t$
  $\Psi_t \leftarrow$ OCEIS(DS_t) ▷ Alg. 7
  $\Pi \leftarrow \Pi \cup \Psi_t$
  if $|\Pi| > s$ **then**
    $\Pi \leftarrow \Pi \setminus \Psi_{t-s}$
    $ADS \leftarrow ADS \setminus DSU_{t-s}$
  end if
end for

Algorithm 7 OCEIS

Input:

LS – learning set
s – maximum size of classifier ensemble
K – maximum number of clusters
CA – clustering algorithm K-means [179]
CM – clusters consistency metric Silhouette Coefficient [165]

Symbols:

$\mathcal{LS}^m$ – minority data
$\mathcal{LS}^M$ – majority data
n – number of minority clusters
N – number of majority clusters
C_k^m – clusters of minority data $\mathcal{LS}^m$
C_k^M – clusters of minority data $\mathcal{LS}^M$
Ψ_i^m – $OCSVM$ model trained on C_i^m cluster
Ψ_j^M – $OCSVM$ model trained on C_j^M cluster
Π^m – minority model set (ensemble)
Π^M – majority model set (ensemble)

Output:

$\Psi(x)$ – Final ensemble with combination rule (Eq. 4.6)

Split LS into minority ($\mathcal{LS}^m$) and majority ($\mathcal{LS}^M$) data
for $k = 1, 2, \ldots, K$ **do**
    $C_k^M \leftarrow$ Create k clusters using CA on $\mathcal{LS}^M$
    $C_k^m \leftarrow$ Create k clusters using CA on $\mathcal{LS}^m$
end for
$n \leftarrow \underset{k=1,2,\ldots,K}{\text{argmax}}\ CM(\mathrm{C}_k^m)$
for $i = 1, 2, \ldots, n$ **do**
    $\Psi_i^m \leftarrow$ Train $OCSVM$ model on C_i^m cluster data
    $\Pi^m \leftarrow \Pi^m \cup \Psi_i^m$
end for
$N \leftarrow \underset{k=1,2,\ldots,K}{\text{argmax}}\ CM(\mathrm{C}_k^M)$
for $j = 1, 2, \ldots, N$ **do**
    $\Psi_j^M \leftarrow$ Train $OCSVM$ model on C_j^M cluster data
    $\Pi^M \leftarrow \Pi^M \cup \Psi_j^M$
end for

It is also worth noting how the drift detector works. As it has already been written, it is a classifier with relearning capabilities. Skipping the first iteration, every chunk data, it calculates the predictive performance of the drift detector classifier ($Score$), which is expressed by the selected metric. This result is compared with the average value ($Score_M$) from the previous data chunks. When the ratio of the current quality and the average from previous iterations is less than the threshold parameter - T, the detector reports the occurrence of drift. In other words, if the current quality falls by a certain percentage compared to the average value, this is treated as a concept drift. Then, all data contained in the accumulated data storage is deleted. In addition, the models are deleted and the average value obtained by the drift detector is set to zero.

There are a few important points to note about the proposed drift detection method. The choice of the $Gmean_s$ for the drift detector and 70% threshold value was determined experimentally in the initial phase of the method design. The idea of using a fixed value was inspired by the work of Klinkenberg and Renz [133], where their drift detector proposal contains a similar scheme using an arbitrarily determined value of the threshold. In addition to the drift detector, the forgetting of the oldest models was also used in this ensemble. It can be argued that *DSE-OC* has a dual drift protection. However, it is important to note that both of these mechanisms perform their respective roles. The forgetting mechanism removes the oldest models from the ensemble, which protect the method from slow and incremental drifts. On the other hand, the drift detector will allow a fast reaction when a sudden concept drift arrives, minimizing the time required to rebuild the whole ensemble. A fixed percentage of the quality drop threshold during detection allows good cooperation between two independently performing techniques and complements each other well to counter different types of concept drifts that occur in non-stationary data streams.

Computation complexity analysis

Let us present the analysis of the time complexity for the *DSE-OC* algorithm. This method combines two different solutions. The final time complexity is the linear combination of several components. First, it is important to determine the appropriate computational complexity for ADS that stores the collected data. Assuming that n_s will denote the number of samples in one data chunk, and S will denote the maximum number of stored data chunks, one can determine the complexity from usage the ADS. Two main operations are performed, retrieving the data and uploading the data. Fetching the entire dataset equals $O(Sn_s)$, since the number of samples and data chunks cannot be exceeded. On the other hand, the complexity of adding a new data chunk is $O(n_s)$, since loading one element into the list [54] has a complexity of $O(1)$. The drift

detector is another factor affecting the time complexity. In this method, the *Multi Layer Perceptron* classifier with one hidden layer and ten neurons was used. The approximate complexity is $O(n_s p^h)$, where h is the number of hidden layers, and p is the number of neurons. Another quite important piece is the method for undersampling the data. Because the complexity is quite dependent on the choice of method that will be used, it can be assumed that the method used will be Instance Hardness Threshold, whose complexity is equal to $O(n_s^2)$ [233]. The method for determining clusters has an equally important influence. This is also a method that can be changed at will and is treated as a parameter. In this case, it can be assumed that the K-means approach will be used to cluster the data, which has a complexity equal to $O(n_s^2)$ [196]. The last element is the complexity of the base classifier. According to the concept of the algorithm the *OCSVM* classifiers is used, which complexity is equal to $O(n_s^3)$ [275]. Summarizing the above analysis, it can be determined that the complexity of the *DSE-OC* method is equal to $O(n_s^3)$, as it is the highest value of all factors.

4.4.1 Experimental evaluation

This section will present the experimental evaluation to check the performance of the method under various imbalanced data streams compared to selected *state-of-the-art* methods. Let us formulate the following research hypotheses:

RQ1: What is the impact of *DSE-OC* on the concept drift in the imbalanced data stream?

RQ2: How flexible is *DSE-OC* to the dynamic imbalance ratio data?

RQ3: What is the predictive performance of *DSE-OC* in comparison to the *state-of-the-art* methods?

Setup

Experiments were performed on various synthetic imbalanced data streams with the presence of concept drifts. The data streams were generated using MOA [24] and stream-learn [150] generators. Tab. 3.3 presents parameters such as a number of objects, chunk size, imbalance ratio, noise level or drift type of generated synthetic data streams. This gives a total of one hundred thousand objects per data stream. The samples had 10 attributes with floating-point values described by labels from two classes – positive and negative. The stream-learn framework was used to generate such diverse data. The proposed method was compared with selected *state-of-the-art* algorithms:

- *REA* – Recursive ensemble approach [51]
- *KMC* – K-means clustering undersampling ensemble [264]
- *L++CDS* – Learn++CDS [62]
- *L++NIE* – Learn++NIE [62]
- *OUSE* – Over and undersampling ensemble [89]
- *MLPC* – Multi-layer perceptron classifier

The testing procedure was based on the test-then-train approach, where each chunk was first used to test and then to learn the model. In this way, it is possible to test almost the entire stream without the problem of duplicate samples. This method of evaluation was provided by the stream learn [150] library. The results were evaluated using 5 selected classification metrics – $Gmean_s$, F_1score, $Precision$, $Recall$, $Specificity$. Statistical analysis was performed on these scores. Pairwise rankings were generated using Wilcoxon rank-sum tests. The project along with the experimental environment has been implemented in Python programming language. The full implementation and results are available on the Github repository[4]. Four selected base classifiers implemented in the scikit-learn [200] library were used:

- *k-Nearest Neighbors* (*KNN*)
- *Support Vector Machine* (*SVM*)
- *Gaussian Naive Bayes* (*GNB*)
- *Decision Tree CART* (*DTC*)

The performed experiments were designed to verify how the proposed method deals with classifying imbalanced data streams with varying characteristics in comparison to selected *state-of-the-art* methods. The results obtained for each data type will be presented below.

Experiment 1 - Dynamic imbalance

The first experiment focused on testing the method using imbalanced data streams with dynamically changing imbalance ratio. In these streams, the class distribution evolves incrementally. Each new data chunk has a different imbalance ratio. Initially, the data stream has 5% positive class objects. Then the number of positive samples increases with

[4]Repository link: https://github.com/w4k2/dse-oc

subsequent data chunks. In about half of the data stream, it reaches 95% of positive class objects. At the same time, the opposite class has 5% of samples. In the following iterations, this ratio returns to the original level, which is 5% of the positive class objects.

Firstly, it is worth paying close attention to the waveform graph in Fig. 4.37. The results are shown as the values of F_1score for particular data chunks. At first glance, it can be seen that for the selected data stream, the proposed method achieves predictive performance at the highest level compared to other methods. The OUSE method in the 75th data chunk experiences some collapse which results in a very strong drop. Apparently, this is a very negative response to the changing imbalance level.

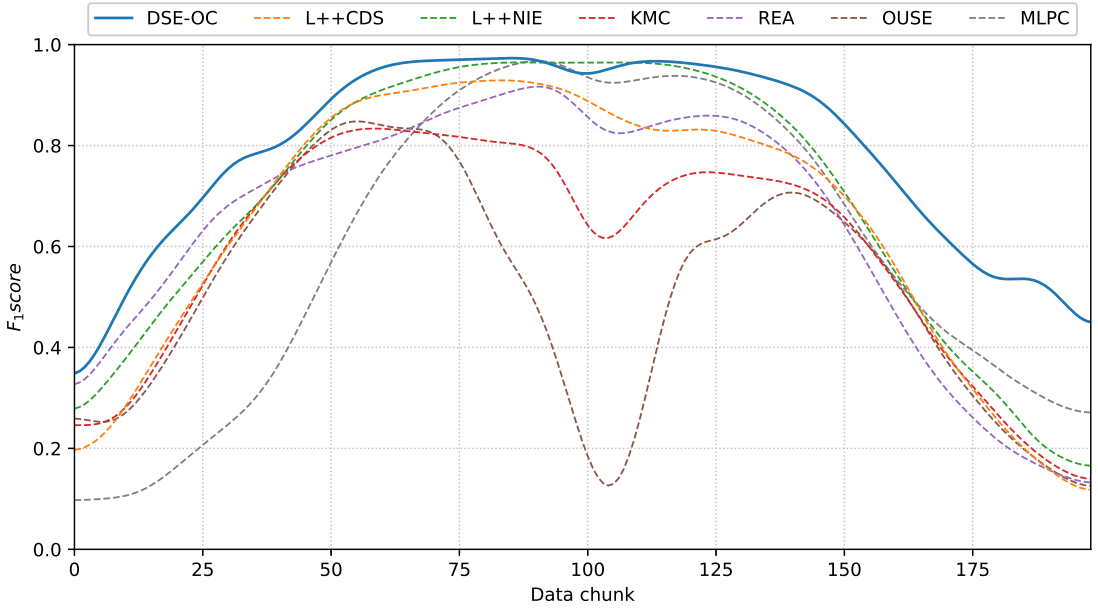


Figure 4.37: *$Gmean_s$ score over the data chunks for synthetic data with sudden drift and dynamic imbalance using GNB base classifier*

Then, the ranking results for these data are presented in Fig. 4.38. Here it is possible to show how the *DSE-OC* method performs compared to other methods but on more data streams. Looking at the rankings for the synthetic data streams with incremental drift and dynamic imbalance, it can be seen that the proposed method obtains very good results for the base classifiers *DTC* and *GNB*. It outperforms the other methods in large majority obtaining a statistically significant advantage. For the other base classifiers, the Learn++CDS method dominates over the *DSE-OC* method. The other methods depending on the metric looked at receive different scores.

For the same data but with sudden drift, the situation is very similar (Fig. 4.39). It is noticeable that slightly more draws occur. *DSE-OC* similarly obtains good results for the *GNB* base classifier and *DTC*. For the *SVM* classifier, half of the compared methods are statistically worse than *DSE-OC*. It can be said that Learn++CDS approach still

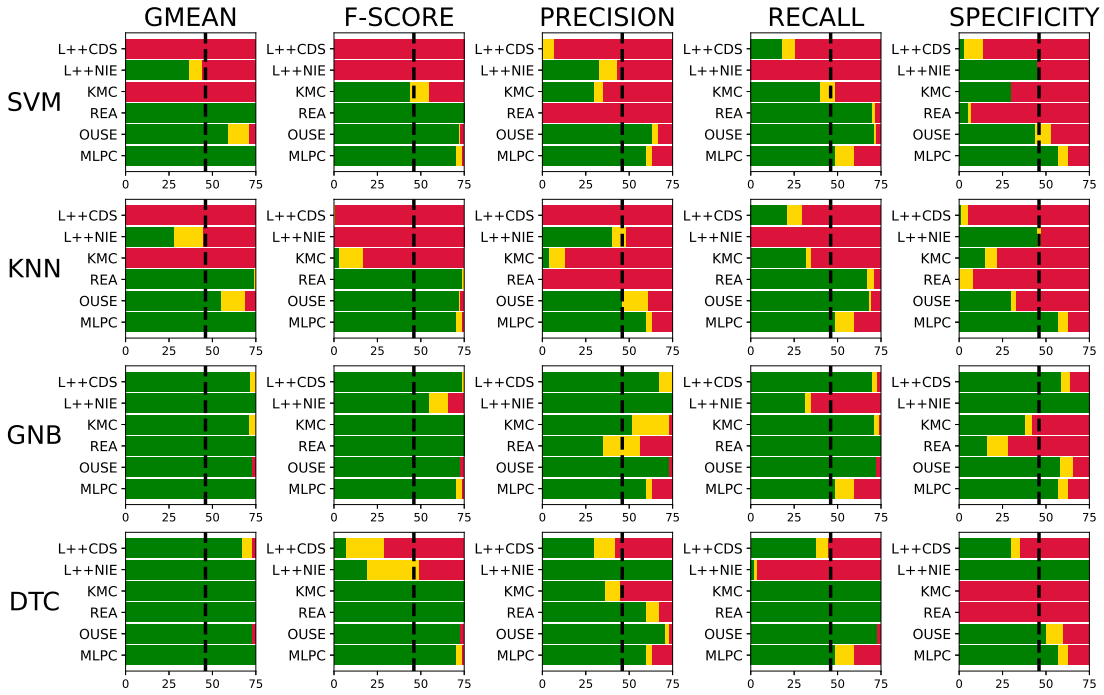


Figure 4.38: *Wilcoxon pair rank-sum tests for synthetic data streams with incremental concept drift and dynamic imbalance ratio. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*

performs best when it comes to *SVM* and *KNN* base classifier. Also occasionally with the *DTC* classifier, it obtains better results in more data streams.

Experiment 2 - Static imbalance

In the next experiment, the data no longer has a dynamic data imbalance ratio. This means that the imbalance is fixed once and remains unchanged until the end of the stream. However, it is still a non-stationary data stream that has a drift concept that occurs in the middle of the stream. This drift concept is once abrupt and once incremental. This is also how the data was divided inside this experiment.

First, it will be described a Fig. 4.40 showing the waveform for a selected data stream that has a static imbalance ratio and a sudden concept drift. In the presented graph it can be seen how well *DSE-OC* performs compared to other tested methods. It is also worth noting that after a drift occurs, the proposed method is able to recover after a few data chunks. This is an essential capability that is useful especially when classifying a data stream with sudden drifts. It is also apparent that not all methods are able to rebuild their predrift performance, and that some methods such as OUSE completely lose their classification rate until nearly the very end of the stream.

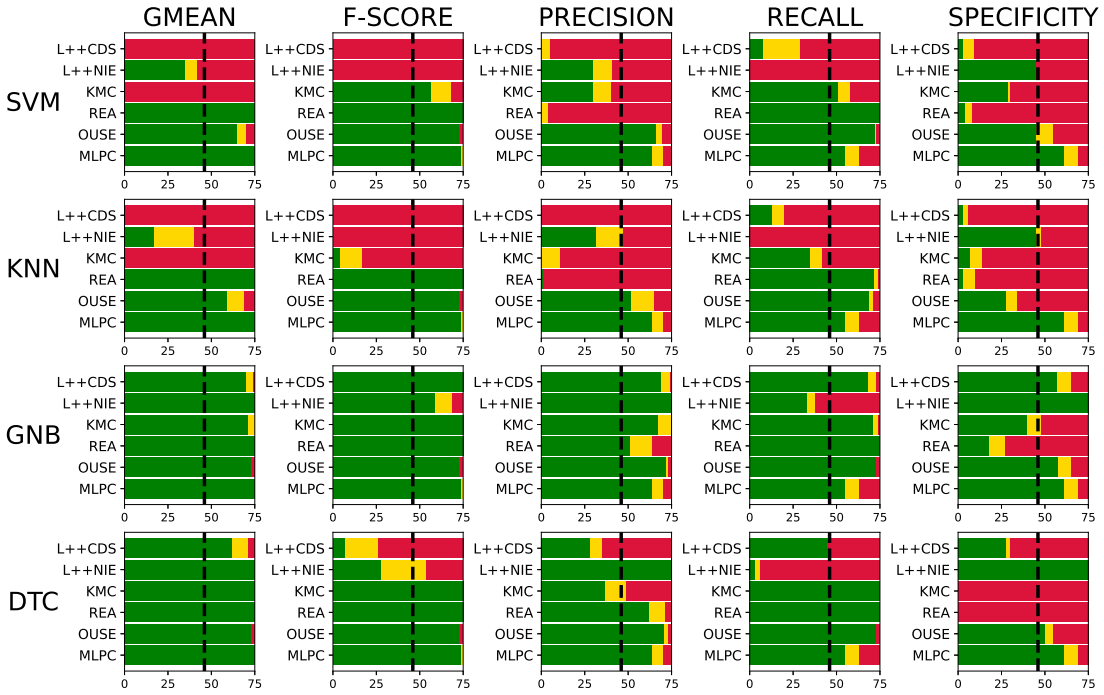


Figure 4.39: *Wilcoxon pair rank-sum tests for synthetic data streams with sudden concept drift and dynamic imbalance ratio. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*

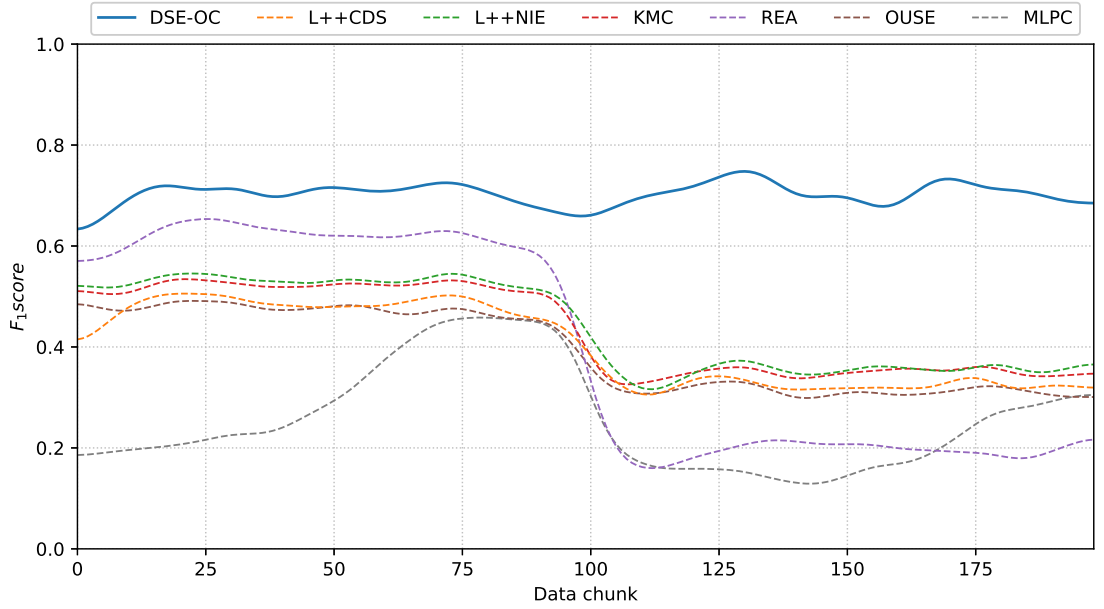


Figure 4.40: *$Gmean_s$ score over the data chunks for synthetic data with sudden drift using GNB base classifier*

Looking at the aggregated results from the ranking tests (Fig. 4.41), the situation is already slightly worse than for the data with dynamic imbalance. Similarly, the best

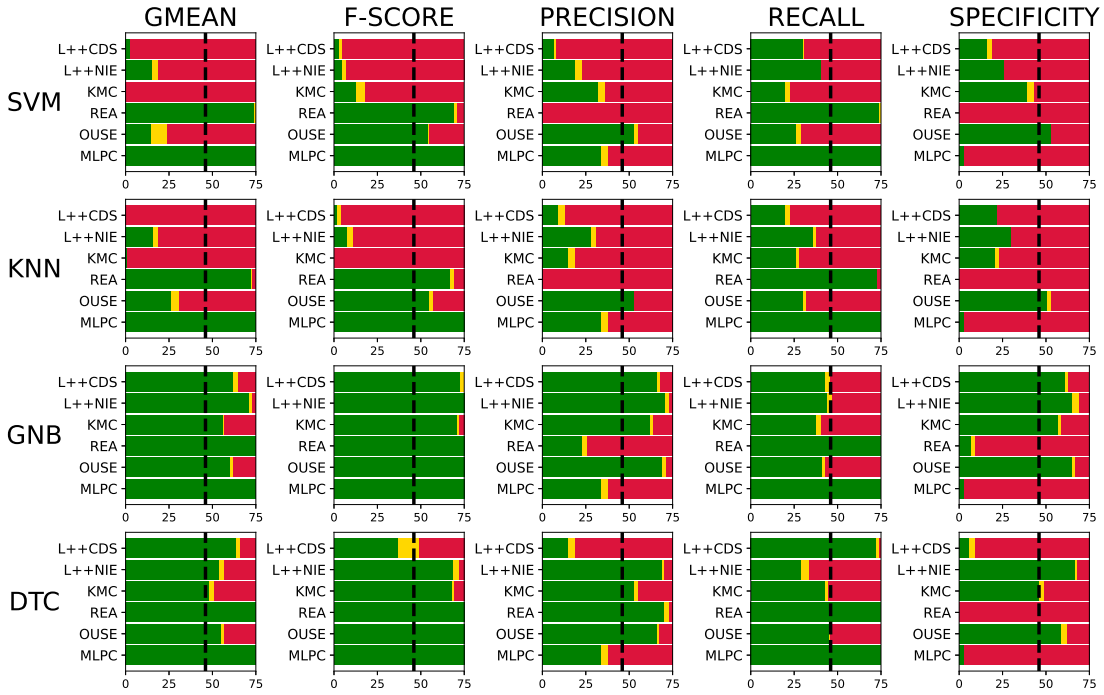


Figure 4.41: *Wilcoxon pair rank-sum tests for synthetic data streams with incremental concept drift. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*

results are obtained for *DTC* and *GNB* classifiers, but not always with statistical significance. For the others base classifiers, it performs worse. However, it obtains more often an advantage for the REA or OUSE method. Looking at the REA results, one can see that this method quite often classifies the data as the minority class. This tendency is strongly reflected in the results, where *Recall* is at a very high level, but with the cost of a low score in *Precision*, *Specificity* or in the aggregated metrics. Similar results are achieved with sudden data streams (Fig. 4.42).

Experiment 3 - Real data

The last experiment is to validate the method using imbalanced real data streams. However, acquiring difficult real data with such demanding characteristics (imbalance and concept drifts) is not quite a simple task. Combining these two problems in a single real-life dataset is rather a rarity. Creating semisynthetic data where an artificial imbalance or concept drift injection is made will cause it to lose its original character and real origin. Therefore, two selected datasets will be used [44]. Their characteristics can be found in the Tab. 3.4.

The first of these "covtypeNorm-1-2vsAll" streams is featured by the occurrence of three concept drifts (Figure). These can be seen around the first 20 chunks, then around the

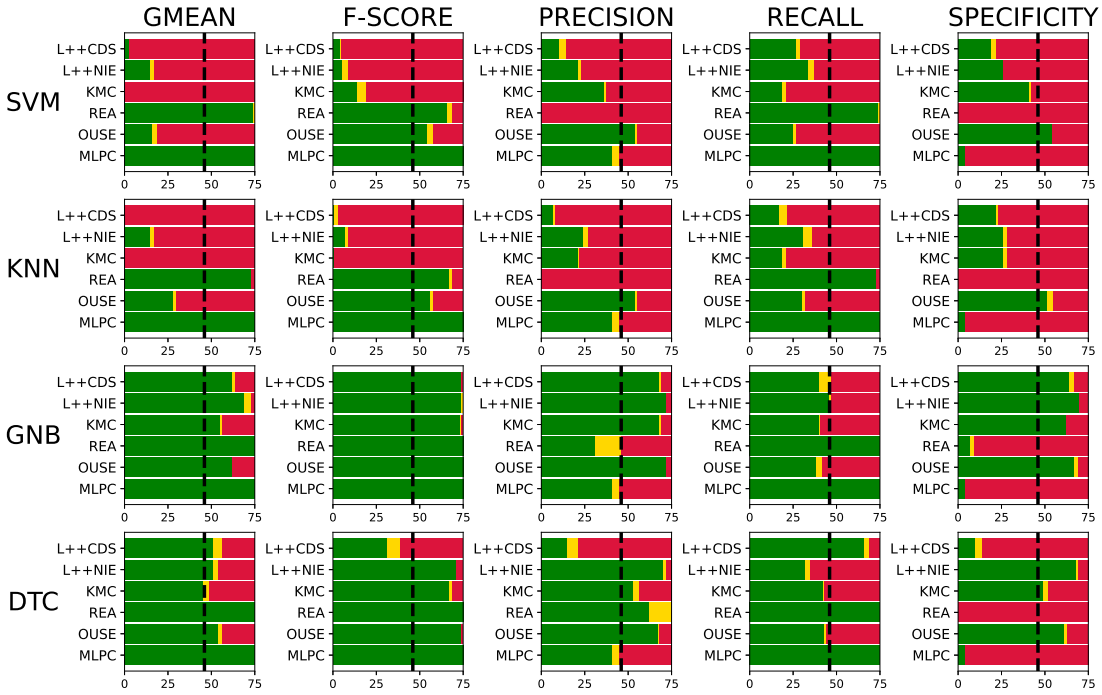


Figure 4.42: *Wilcoxon pair rank-sum tests for synthetic data streams with sudden concept drift. Dashed vertical line is a critical value with a significance level 0.05 (green – win, yellow – tie, red – loss)*

60th chunk of data and between the 100th and 120th chunk. From the waveform plot, it can be seen that the proposed method solves this problem with not too bad results. At the first drift, the *DSE-OC* classification quality performs quite poorly. Then in subsequent data chunks the prediction improves and reaches a level comparable to other methods. The next drift causes a crucial drop. However, this is rebuilt with subsequent iterations of the data stream. *DSE-OC* performs quite well compared to other methods, achieving similar performance.

The results obtained by the proposed method on the second stream "poker-lsn-1-2vsAll" at first glance seem to be much better compared to the other algorithms (Fig. 4.44). It is very clear that *DSE-OC* dominates for most of the data stream and only in some moments its predictive performance falls below Learn++NIE. The worst result is obtained by the REA method, which apparently cannot manage this data stream. It is also possible that this stream is much more difficult to classify. Compared to the previous one, because the highest score is around 0.40, and for the stream "covtypeNorm" most of the methods at some point get a score over 0.80. Better results indicate that the ability of *DSE-OC* to classify difficult data problems is significantly better.

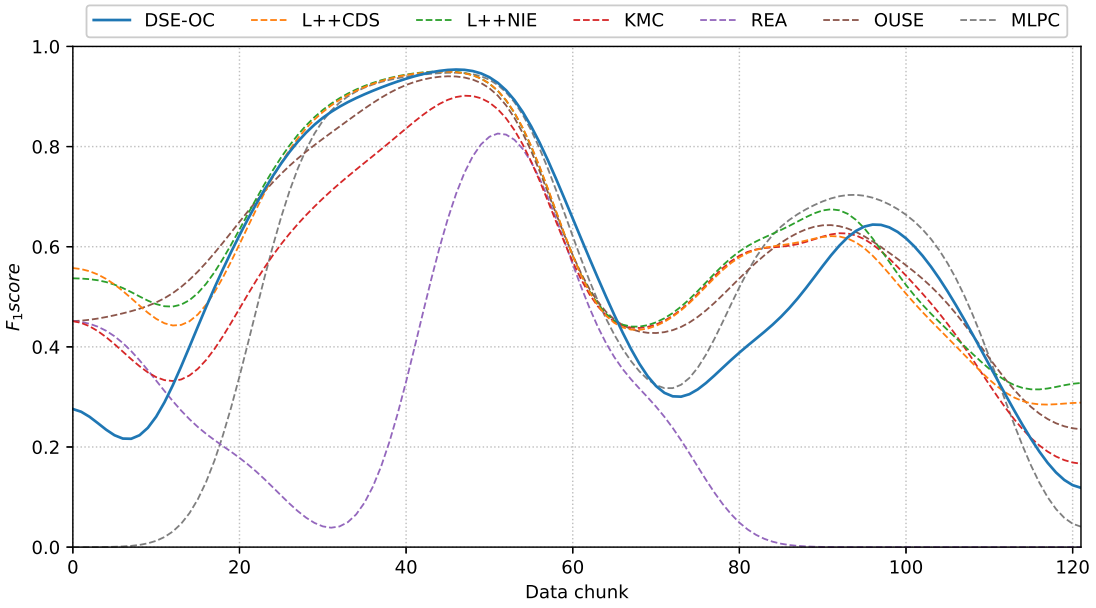


Figure 4.43: *F_1score over the data chunks for "covtypeNorm-1-2vsAll" using GNB base classifier*

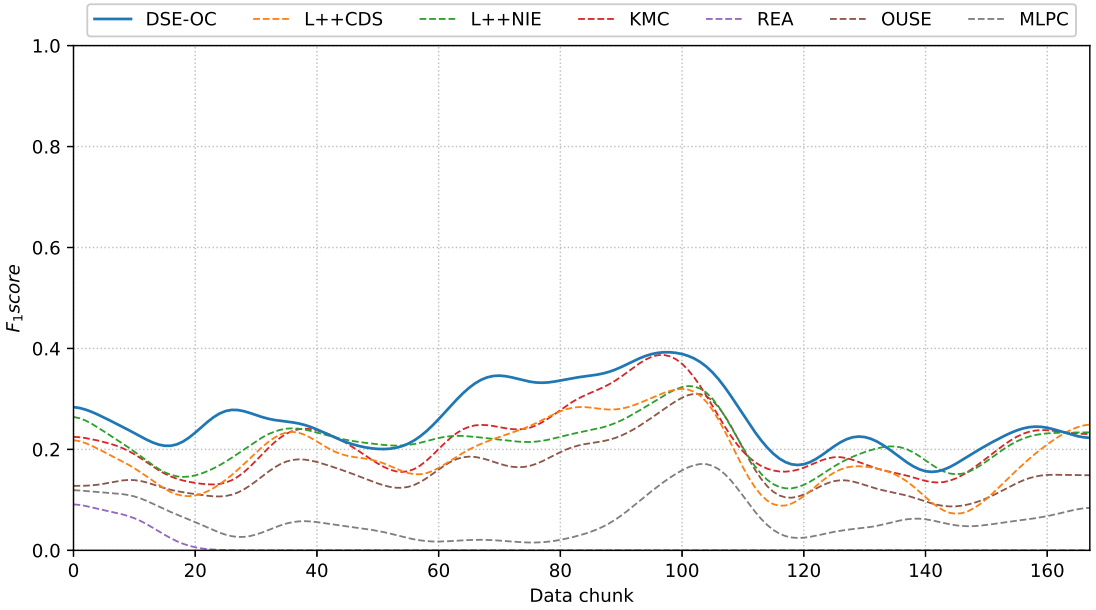


Figure 4.44: *F_1score over the data chunks for "poker-lsn-1-2vsAll" using GNB base classifier*

4.4.2 Lessons learned

To summarize the above analysis of the obtained results, answers to the research questions stated earlier will be formulated:

RQ1: How flexible is DSE-OC to the dynamic imbalance ratio data?

The performed experiments on a large collection of data streams with dynamic imbalance ratio supported by statistical analysis allows to conclude that the proposed method is robust to this type of data. This seems to be proven by the fact that for the static imbalanced data, the quality is noticeably worse than for the dynamically imbalanced data. Furthermore, compared to most *state-of-the-art* methods, the results are statistically better for dynamically imbalanced streams. The performed tests lead to the conclusion that the proposed method performs well on this type of data.

RQ2: What is the impact of DSE-OC on the concept drift in the imbalanced data stream?

It is obvious that the concept drift has a very negative effect on the proposed method. However, some important facts must be noted. *DSE-OC* has a very good ability to recover from a drift. This is particularly evident in the concept drift plots, where one can see how the proposed method can return to the predictive performance of the previous concept. Unfortunately, it cannot be assumed that such a situation is fully repeatable because it also depends on how challenging the new concept is. It is possible to conclude that *DSE-OC* has a good potential to deal with concept drift.

RQ3: What is the predictive performance of DSE-OC in comparison to the* state-of-the-art *methods?

Compared to other selected *state-of-the-art* methods, *DSE-OC* does not appear to be the worst. Likewise, to say that it is the best method in every situation would be a complete mistake. However, it is worth noting that in some situations – *DTC* or *GNB* base classifier and dynamic imbalance – the results are very satisfactory and better with statistical significance. The situation is similar for real origin data, where for two selected streams once *DSE-OC* dominates and in the other stream it performs at a medium level. Nevertheless, in answer to the stated research question, it must be clearly noted that the method obtains good results compared to selected *state-of-the-art* methods.

Chapter 5

Conclusions and future research directions

This dissertation focused on solving binary imbalanced data stream classification problems with concept drift using hybrid sampling techniques with data accumulation and one-class models. The conducted research and the obtained results provide an unambiguous conclusion that the proposed approaches have considerable potential, and the results obtained by the invented methods often equal or outperform the selected *state-of-the-art* algorithms. The posed research hypothesis:

One may design ensemble methods that use data sampling techniques and one-class classifiers, which can outperform state-of-the-art ensemble algorithms for imbalanced data streams classification.

has been to be proved by the realized research objectives:

- **Designing the one-class classifiers ensemble method to solve the problem of the imbalanced data stream classification.**

  This objective was achieved by proposition of the imbalanced data streams classification algorithm based on the one-class classifier ensemble - *OCEIS*. The method achieves results at a similar level to the compared methods, but it is worth noticing that it performs best on real stream data, which is its important advantage. Another advantage is that there is no tendency towards the excessive classification of objects from one of the classes. Such "stability" contributes significantly to improving the quality of classification and obtaining satisfactory results.

  Proposition of *OCEIS* was published in [132].

– **Extending the one-class classifiers ensemble method by introducing the improved weighted decision rule for better adaptation to imbalanced data streams.**

  This objective was achieved by designing the novel method for the classification of imbalanced data streams - *OCWE*. It is the method based on one-class classification ensembles, extending the original idea from the *OCEIS* approach [132]. Introduced modifications and extensive research resulted in some improvements and effectiveness of classifying imbalanced data streams. Compared to other *state-of-the-art* methods, *OCWE* is not the worst one and can obtain predictive performance at a similar level to other methods. Under some conditions, *OCWE* achieves better performance. This method is ideally suited for the classification of strongly imbalanced data. The multiplicity of parameters provides a good adaptation ability to the currently handled problem.

– ***Proposing the classifier for imbalanced data streams with hybrid data accumulation sampling technique.***

  This objective was achieved by developing method for imbalanced stream classification. *DSC* performed favorably in comparison with other dedicated algorithms. The evaluation of the predictive abilities of the techniques was conducted on the basis of computer experiments. The algorithm utilizes memory buffer in order to propagate the instances from the previous data chunks that were chosen as the representatives. Since the buffer is fixed size, after it is full some instances must be removed from it. In the current implementation, the oldest examples are deleted.

  Proposition of *DSC* was published in [28]

– ***Upgrading the classifier with data accumulation hybrid sampling to the weighted bagging ensemble for imbalanced data stream classification.***

  This objective was achieved by introducing the new method for imbalanced data stream classification. *DSCB* develops the idea of *DSC* algorithm with the solution that is a kind of bagging. The proposed *DSCB* method is basically a set of classifiers. The main idea focuses on creating by weighted bagging new models using data that have been accumulated from previous chunks. The proposed approach obtains good results, which can be verified by the performed comparative analysis with *state-of-the-art* methods.

– ***Improving the imbalanced data stream classifier with hybrid data accumulation sampling to the ensemble method with concept drift detector.***

  This objective was achieved by proposition of the novel, effective classifier ensemble method couple with a preprocessing framework for a drifting imbalanced data

stream classification task. The main idea of *DSE* bases on the accumulation of selected samples from previous chunks for later strengthening and balancing the data. It allows for specific oversampling that does not require generating artificial data. The research conducted on a wide range of real and computer-generated data streams confirmed the effectiveness of the proposed solution. It highlighted its strengths in comparison with *state-of-the-art* methods and its high robustness to the label noise.

– ***Designing the ensemble method for imbalanced data stream classification that combines hybrid data accumulation sampling and one-class classifiers.***

This objective was achieved by establishing the new method for imbalanced data stream classification - *DSE-OC*, which integrates two different approaches. It is the combination of best features from *DSE* and *OCEIS* propositions. In one method there is at the same time an adaptation of binary problems for classification using one class SVM models with a method that accumulates data for semi-synthetic oversampling of imbalanced data. The performed tests concluded that the presented method has a reasonably good ability to classify imbalanced data streams. The evaluation including statistical comparative analysis showed that the method repeatedly gains a better result with statistical significance than selected *state-of-the-art* methods.

Publications:

– Klikowski, Jakub, and Michał Woźniak. *"Multi sampling random subspace ensemble for imbalanced data stream classification."* International Conference on Computer Recognition Systems, pages 360-369, Springer, Cham, 2019.

– Klikowski, Jakub, Paweł Ksieniewicz, and Michał Woźniak. *"A genetic-based ensemble learning applied to imbalanced data classification."* International Conference on Intelligent Data Engineering and Automated Learning, pages 340-352, Springer, Cham, 2019.
CORE C, MNISW 20

– Bobowska, Barbara, Jakub Klikowski, and Michał Woźniak. *"Imbalanced Data Stream Classification Using Hybrid Data Preprocessing."* Workshop in Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 402-413, Springer, Cham, 2019.
CORE A, MNISW 140

- Klikowski, Jakub *"Ensemble data preprocessing based methods for imbalanced data stream classification"* - extended abstract, PP-RAI'2019, pages 53-55. ISBN 978-83-943803-2-8

- Klikowski, Jakub, and Michał Woźniak. *"Employing One-Class SVM Classifier Ensemble for Imbalanced Data Stream Classification."* International Conference on Computational Science, pages 117-127, Springer, Cham, 2020.
  CORE A, MNISW 140

- Grzyb, Joanna, Jakub Klikowski, and Michał Woźniak. *"Hellinger Distance Weighted Ensemble for imbalanced data stream classification."* Journal of Computational Science, Volume 51: 101314, 2021.
  IF 3.976, MNISW 100

- Klikowski, Jakub, and Robert Burduk. *"Clustering and Weighted Scoring Algorithm Based on Estimating the Number of Clusters"*, International Conference on Computational Science, pages 40-49, Springer, Cham, 2021.
  CORE A, MNISW 140

- Klikowski, Jakub, and Robert Burduk. *"Distance Metrics in Clustering and Weighted Scoring Algorithm"* International Conference on Computer Recognition Systems, pages 23-33, Springer, Cham, 2021.

Under review:

- Klikowski, Jakub, and Michał Woźniak. *"One-Class Weighted Ensemble for Drifted Imbalanced Data Stream Classification"*, Engineering Applications of Artificial Intelligence (Sec. 3.2)
  IF 6.212, MNISW 100

- Klikowski, Jakub, and Michał Woźniak. *Deterministic Sampling Classifier with Weighted Bagging for Drifted Imbalanced Data Stream Classification*, Applied Soft Computing (Sec. 4.2)
  IF 6.725, MNISW 200

Future works

Further research development on the methods and problems presented in the dissertation can be extended to:

- The proposition of new drift detector that can be implemented in DSE and DSE-OC methods. It is worthwhile to consider a drift detector technique that does not rely on the use of predictive performance measures.

- The automatic cluster number selection technique used in the OCEIS, OCWE, and DSE-OC methods should be rebuilt in such a way that it does not perform a brute force search in a limited area, but selects the best number of clusters using intelligent algorithms.
- All proposed methods have the potential to extend research to multiclass problems. Special attention should be paid to methods based on one-class classifiers, which could be very easily adapted to multiclass problems without using decomposition techniques.
- From the perspective of difficult data, which was the main topic of the dissertation, an interesting direction of development would be to try to use the proposed methods for problems that do not have full data labelling, employing additionally different active learning approaches.
- An essential part of future development will be to expand the pool of real imbalanced data streams. However, as mentioned earlier, there is currently not much availability for this type of real origin data.

Acknowledgement

This work was supported by the Polish National Science Centre under the grant No. 2017/27/B/ST6/01325.

Bibliography

[1] Abdiansah Abdiansah and Retantyo Wardoyo. Time complexity analysis of support vector machines (svm) in libsvm. *International journal computer and application*, 128(3):28–34, 2015.

[2] Salina Adinarayana and E Ilavarasan. An efficient approach for opinion mining from skewed twitter corpus using over sampled imbalance data learning. In *2017 International Conference on Intelligent Communication and Computational Techniques (ICCT)*, pages 42–47. IEEE, 2017.

[3] Charu C. Aggarwal. *Data Classification*, pages 285–344. Springer International Publishing, Cham, 2015.

[4] Charu C Aggarwal, Xiangnan Kong, Quanquan Gu, Jiawei Han, and S Yu Philip. Active learning: A survey. In *Data Classification: Algorithms and Applications*, pages 571–605. CRC Press, 2014.

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

[6] Aida Ali, Siti Mariyam Shamsuddin, and Anca L Ralescu. Classification with class imbalance problem. *Int. J. Advance Soft Compu. Appl*, 5(3), 2013.

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

[8] Mélodie Angeletti, Jean-Marie Bonny, and Jonas Koko. Parallel euclidean distance matrix computation on big datasets. 2019.

[9] Dana Angluin and Philip Laird. Learning from noisy examples. *Mach. Learn.*, 2(4):343–370, April 1988.

[10] Manuel Baena-Garcıa, José del Campo-Ávila, Raúl Fidalgo, Albert Bifet, R Gavalda, and R Morales-Bueno. Early drift detection method. In *Fourth International Workshop on Knowledge Discovery from Data Streams*, volume 6, pages 77–86, 2006.

[11] Ishwar Baidari and Nagaraj Honnikoll. Accuracy weighted diversity-based online boosting. *Expert Systems with Applications*, 160:113723, 2020.

[12] Roberto SM Barros, Danilo RL Cabral, Paulo M Gonçalves Jr, and Silas GTC Santos. Rddm: Reactive drift detection method. *Expert Systems with Applications*, 90:344–355, 2017.

[13] Roberto Souto Maior Barros and Silas Garrido T Carvalho Santos. A large-scale comparison of concept drift detectors. *Information Sciences*, 451:348–370, 2018.

[14] Roberto Souto Maior de Barros, Silas Garrido T. de Carvalho Santos, and Paulo Mauricio Gonçalves Júnior. A boosting-like online learning ensemble. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1871–1878, 2016.

[15] Aliaksandr Barushka and Petr Hajek. Spam filtering using integrated distribution-based balancing approach and regularized deep neural networks. *Applied Intelligence*, 48(10):3538–3556, 2018.

[16] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29, June 2004.

[17] Gustavo EAPA Batista, Ana LC Bazzan, and Maria Carolina Monard. Balancing training data for automated annotation of keywords: a case study. In *WOB*, pages 10–18, 2003.

[18] Mohammad Beheshti Roui, Mariam Zomorodi, Masoomeh Sarvelayati, Moloud Abdar, Hamid Noori, Paweł Pławiak, Ryszard Tadeusiewicz, Xujuan Zhou, Abbas Khosravi, Saeid Nahavandi, and U. Rajendra Acharya. A novel approach based on genetic algorithm to speed up the discovery of classification rules on gpus. *Knowledge-Based Systems*, 231:107419, 2021.

[19] Mohamed Bekkar, Hassiba Kheliouane Djemaa, and Taklit Akrouf Alitouche. Evaluation measures for models assessment over imbalanced data sets. *Journal of Information Engineering and Applications*, 3(10), 2013.

[20] Richard E Bellman. *Adaptive control processes*. Princeton university press, 2015.

[21] Peter Bellmann, Patrick Thiam, and Friedhelm Schwenker. Multi-classifier-systems: architectures, algorithms and applications. In *Computational Intelligence for Pattern Recognition*, pages 83–113. Springer, 2018.

[22] Sven Berg. Condorcet's jury theorem, dependency among jurors. *Social Choice and Welfare*, 10(1):87–95, 1993.

[23] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 443–448. SIAM, 2007.

[24] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *Journal of Machine Learning Research*, 11(May):1601–1604, 2010.

[25] Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Leveraging bagging for evolving data streams. In *Machine Learning and Knowledge Discovery in Databases*, pages 135–150, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[26] Albert Bifet, Gianmarco De Francisci Morales, Jesse Read, Geoff Holmes, and Bernhard Pfahringer. Efficient online evaluation of big data stream classifiers. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 59–68, 2015.

[27] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, address = Berlin, Heidelberg, 2006.

[28] Barbara Bobowska, Jakub Klikowski, and Michał Woźniak. Imbalanced data stream classification using hybrid data preprocessing. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 402–413. Springer, 2019.

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

[30] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. Routledge, Boca Raton, 1984.

[31] Erica Briscoe and Jacob Feldman. Conceptual complexity and the bias/variance tradeoff. *Cognition*, 118(1):2–16, 2011.

[32] Gavin Brown. Ensemble learning. *Encyclopedia of Machine Learning*, 312:15–19, 2010.

[33] Gavin Brown and Ludmila I. Kuncheva. "good" and "bad" diversity in majority vote ensembles. In *Multiple Classifier Systems*, pages 124–133, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[34] Håkan Brunzell and Jonny Eriksson. Feature reduction for classification of multidimensional data. *Pattern Recognition*, 33(10):1741–1748, 2000.

[35] Dariusz Brzezinski, Leandro L Minku, Tomasz Pewinski, Jerzy Stefanowski, and Artur Szumaczuk. The impact of data difficulty factors on classification of imbalanced and concept drifting data streams. *Knowledge and Information Systems*, pages 1–41, 2021.

[36] Dariusz Brzeziński and Jerzy Stefanowski. Accuracy updated ensemble for data streams with concept drift. In *Hybrid Artificial Intelligent Systems*, pages 155–163, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[37] Dariusz Brzezinski and Jerzy Stefanowski. Combining block-based and online methods in learning ensembles from concept drifting data streams. *Information Sciences*, 265:50–67, 2014.

[38] Dariusz Brzezinski and Jerzy Stefanowski. Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):81–94, 2014.

[39] 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, 2020.

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

[41] Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Advances in Knowledge Discovery and Data Mining*, pages 475–482, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

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

[43] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.

[44] Alberto Cano and Bartosz Krawczyk. Kappa updated ensemble for drifting data stream mining. *Machine Learning*, 109(1):175–218, 2020.

[45] Rodolfo C Cavalcante, Rodrigo C Brasileiro, Victor LF Souza, Jarley P Nobrega, and Adriano LI Oliveira. Computational intelligence and financial markets: A survey and future directions. *Expert Systems with Applications*, 55:194–211, 2016.

[46] Joseph Chee Chang, Saleema Amershi, and Ece Kamar. Revolt: Collaborative crowdsourcing for labeling machine learning datasets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, page 2334–2346, New York, NY, USA, 2017.

[47] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[48] 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.

[49] Nitesh V. Chawla, Aleksandar Lazarevic, Lawrence O. Hall, and Kevin W. Bowyer. *SMOTEBoost: Improving Prediction of the Minority Class in Boosting*, pages 107–119. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[50] Sheng Chen and Haibo He. SERA: Selectively recursive approach towards nonstationary imbalanced stream data mining. In *2009 International Joint Conference on Neural Networks*, pages 522–529, 2009.

[51] 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.

[52] Sheng Chen, Haibo He, Kang Li, and Sachi Desai. MuSeRa: Multiple selectively recursive approach towards imbalanced stream data mining. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2010.

[53] Fang Chu and Carlo Zaniolo. Fast and light boosting for adaptive mining of data streams. In *Advances in Knowledge Discovery and Data Mining*, pages 282–292, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[54] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT Press, 2009.

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

[56] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979.

[57] Roberto Souto Maior de Barros and Silas Garrido T. de Carvalho Santos. An overview and comprehensive comparison of ensembles for concept drift. *Information Fusion*, 52:213–244, 2019.

[58] Silas Garrido Teixeira de Carvalho Santos, Paulo Mauricio Gonçalves Júnior, Geyson Daniel dos Santos Silva, and Roberto Souto Maior de Barros. Speeding up recovery from concept drifts. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 179–194. Springer, 2014.

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

[60] Chesner Désir, Simon Bernard, Caroline Petitjean, and Laurent Heutte. One class random forests. *Pattern Recognition*, 46(12):3490–3506, 2013.

[61] Thomas G Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1994.

[62] Gregory Ditzler and Robi Polikar. Incremental learning of concept drift from streaming imbalanced data. *IEEE transactions on knowledge and data engineering*, 25(10):2283–2301, 2012.

[63] Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 155–164, 1999.

[64] Pedro Domingos. A unified bias-variance decomposition. In *Proceedings of 17th International Conference on Machine Learning*, pages 231–238, 2000.

[65] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[66] Molla S Donaldson, Janet M Corrigan, Linda T Kohn, et al. *To err is human: building a safer health system*, volume 6. National Academies Press, 2000.

[67] Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. A survey on ensemble learning. *Frontiers of Computer Science*, pages 1–18.

[68] David L Donoho et al. High-dimensional data analysis: The curses and blessings of dimensionality. *AMS Conference on Mathematical Challenges of the 21st Century*, 1(2000):1–33, 2000.

[69] Anton Dries and Ulrich Rückert. Adaptive concept drift detection. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 2(5-6):311–327, 2009.

[70] Kaibo Duan, S Sathiya Keerthi, Wei Chu, Shirish Krishnaj Shevade, and Aun Neow Poo. Multi-category classification by soft-max combination of binary classifiers. In

International Workshop on Multiple Classifier Systems, pages 125–134. Springer, 2003.

[71] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, USA, 2000.

[72] R.P.W. Duin. The combining classifier: to train or not to train? In *2002 International Conference on Pattern Recognition*, volume 2, pages 765–770 vol.2, 2002.

[73] Ryan Elwell and Robi Polikar. Incremental learning in nonstationary environments with controlled forgetting. In *2009 International Joint Conference on Neural Networks*, pages 771–778, 2009.

[74] Ryan Elwell and Robi Polikar. Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, 22(10):1517–1531, 2011.

[75] Sarah M. Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recognition*, 58:121–134, 2016.

[76] Sandra Garcia Esparza, Michael P O'Mahony, and Barry Smyth. Mining the real-time web: a novel approach to product recommendation. *Knowledge-Based Systems*, 29:3–11, 2012.

[77] Ludwig Fahrmeir, Thomas Kneib, Stefan Lang, and Brian Marx. *Regression Models*, pages 21–72. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[78] Wei Fan, Salvatore J. Stolfo, Junxin Zhang, and Philip K. Chan. AdaCost: Misclassification cost-sensitive boosting. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, page 97–105, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[79] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C Prati, Bartosz Krawczyk, and Francisco Herrera. *Learning from imbalanced data sets*. Springer, 2018.

[80] Alberto Fernández, Salvador Garcıa, 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.

[81] Alberto Fernández, Victoria López, Mikel Galar, María José del Jesus, and Francisco Herrera. Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowledge-Based Systems*, 42:97–110, 2013.

[82] Oscar Fontenla-Romero, Bertha Guijarro-Berdiñas, Beatriz Pérez-Sánchez, and Amparo Alonso-Betanzos. A new convex objective function for the supervised learning of single-layer neural networks. *Pattern Recognition*, 43(5):1984–1992, 2010.

[83] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, May 2014.

[84] Isvani Frias-Blanco, José del Campo-Ávila, Gonzalo Ramos-Jimenez, Rafael Morales-Bueno, Agustin Ortiz-Diaz, and Yailé Caballero-Mota. Online and non-parametric drift detection methods based on hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):810–823, 2014.

[85] Mohamed Medhat Gaber. Advances in data stream mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):79–85, 2012.

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

[87] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer, 2004.

[88] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4), March 2014.

[89] Jing Gao, Bolin Ding, Wei Fan, Jiawei Han, and S Yu Philip. Classifying data streams with skewed class distributions and concept drifts. *IEEE Internet Computing*, 12(6), 2008.

[90] Jing Gao, Wei Fan, Jiawei Han, and Philip S Yu. A general framework for mining concept-drifting data streams with skewed distributions. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 3–14. SIAM, 2007.

[91] Salvador Garca, Julin Luengo, and Francisco Herrera. *Data Preprocessing in Data Mining*. Springer Publishing Company, Incorporated, 2014.

[92] Luís PF Garcia, André CPLF de Carvalho, and Ana C Lorena. Effect of label noise in the complexity of classification problems. *Neurocomputing*, 160:108–119, 2015.

[93] Vicente García, José Salvador Sánchez, and Ramón Alberto Mollineda. On the effectiveness of preprocessing methods when dealing with different levels of class imbalance. *Knowledge-Based Systems*, 25(1):13–21, 2012.

[94] Adel Ghazikhani, Reza Monsefi, and Hadi Sadoghi Yazdi. Ensemble of online neural networks for non-stationary and imbalanced data streams. *Neurocomputing*, 122:535–544, 2013.

[95] Adel Ghazikhani, Reza Monsefi, and Hadi Sadoghi Yazdi. Recursive least square perceptron model for non-stationary and imbalanced data stream classification. *Evolving Systems*, 4(2):119–131, 2013.

[96] Giorgio Giacinto and Fabio Roli. Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing*, 19(9-10):699–707, 2001.

[97] Heitor M Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9):1469–1495, 2017.

[98] Heitor Murilo Gomes, Jean Paul Barddal, Fabrício Enembreck, and Albert Bifet. A survey on ensemble learning for data stream classification. *ACM Computing Surveys (CSUR)*, 50(2):1–36, 2017.

[99] Mabel González, Christoph Bergmeir, Isaac Triguero, Yanet Rodríguez, and José M Benítez. Self-labeling techniques for semi-supervised time series classification: an empirical study. *Knowledge and Information Systems*, 55(2):493–528, 2018.

[100] Nico Görnitz, Luiz Alberto Lima, Klaus-Robert Müller, Marius Kloft, and Shinichi Nakajima. Support vector data descriptions and k-means clustering: One class? *IEEE Transactions on Neural Networks and Learning Systems*, 29(9):3994–4006, 2017.

[101] Nizar Grira, Michel Crucianu, and Nozha Boujemaa. Unsupervised and semi-supervised clustering: a brief survey. *A Review of Machine Learning Techniques for Processing Multimedia Content*, 1:9–16, 2004.

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

[103] Li Guo, Samia Boukir, and Nesrine Chehata. Support vectors selection for supervised learning using an ensemble approach. In *2010 20th International Conference on Pattern Recognition*, pages 37–40. IEEE, 2010.

[104] Shivani Gupta and Atul Gupta. Dealing with noise problem in machine learning data-sets: A systematic review. *Procedia Computer Science*, 161:466–474, 2019.

[105] Hui Han, Wenyuan Wang, and Binghuan Mao. Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In *Advances in Intelligent Computing, International Conference on Intelligent Computing, ICIC 2005, Hefei, China, August 23-26, 2005, Proceedings, Part I*, pages 878–887, 2005.

[106] 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.

[107] Stevan Harnad. The annotation game: On turing (1950) on computing, machinery, and intelligence. *The Turing Test Sourcebook: Philosophical and Methodological Issues in the Quest for the Thinking Computer*, 2006.

[108] P. Hart. The condensed nearest neighbor rule (corresp.). *IEEE Transactions on Information Theory.*, 14(3):515–516, September 2006.

[109] Trevor Hastie, Jerome Friedman, and Robert Tibshirani. *Unsupervised Learning*, pages 437–508. Springer New York, New York, NY, 2001.

[110] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, USA, 3rd edition, 2008.

[111] Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li. ADASYN: adaptive synthetic sampling approach for imbalanced learning. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2008, part of the IEEE World Congress on Computational Intelligence, WCCI 2008, Hong Kong, China, June 1-6, 2008*, pages 1322–1328, 2008.

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

[113] Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10456–10465. Curran Associates, Inc., 2018.

[114] Mauricio A. Hernández and Salvatore J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery*, 2(1):9–37, January 1998.

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

[116] Bao-Gang Hu and Wei-Ming Dong. A study on cost behaviors of binary classification measures in class-imbalanced problems. *arXiv preprint arXiv:1403.7100*, 2014.

[117] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

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

[119] Sarah Itani, Fabian Lecron, and Philippe Fortemps. A one-class classification decision tree based on kernel density estimation. *Applied Soft Computing*, 91:106250, 2020.

[120] Ali Haghpanah Jahromi and Mohammad Taheri. A non-parametric mixture of gaussian naive bayes classifiers based on local independent features. In *2017 Artificial Intelligence and Signal Processing Conference (AISP)*, pages 209–212. IEEE, 2017.

[121] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. Classification. In *An Introduction to Statistical Learning: with Applications in R*, pages 127–173. Springer New York, New York, NY, 2013.

[122] Janardan and Shikha Mehta. Concept drift in streaming data classification: Algorithms, platforms and issues. *Procedia Computer Science*, 122:804–811, 2017. 5th International Conference on Information Technology and Quantitative Management, ITQM 2017.

[123] Rachsuda Jiamthapthaksin, Jiyeon Choo, Chun-sheng Chen, Oner Ulvi Celepcikay, Christian Giusti, and Christoph F Eick. Mosaic: Agglomerative clustering with gabriel graphs. In *Complex Data Warehousing and Knowledge Discovery for Advanced Retrieval Development: Innovative Methods and Applications*, pages 231–250. IGI Global, 2010.

[124] Hao Jiang, Xiaojie Qiu, Jing Chen, Xinyu Liu, Xiren Miao, and Shengbin Zhuang. Insulator fault detection in aerial images based on ensemble learning with multilevel perception. *IEEE Access*, 7:61797–61810, 2019.

[125] Thorsten Joachims and Filip Radlinski. Search engines that learn from implicit feedback. *Computer*, 40(8):34–40, 2007.

[126] João Roberto Bertini Junior and Maria do Carmo Nicoletti. An iterative boosting-based ensemble for streaming data classification. *Information Fusion*, 45:66–78, 2019.

[127] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[128] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, January 1994.

[129] Samina Khalid, Tehmina Khalil, and Shamila Nasreen. A survey of feature selection and feature extraction techniques in machine learning. In *2014 Science and Information Conference*, pages 372–378, 2014.

[130] Shehroz S Khan and Amir Ahmad. Relationship between variants of one-class nearest neighbors and creating their accurate ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1796–1809, 2018.

[131] Shehroz S. Khan and Michael G. Madden. A survey of recent trends in one class classification. In *Artificial Intelligence and Cognitive Science*, pages 188–197, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[132] Jakub Klikowski and Michał Woźniak. Employing one-class svm classifier ensemble for imbalanced data stream classification. In *International Conference on Computational Science*, pages 117–127. Springer, 2020.

[133] Ralf Klinkenberg and Ingrid Renz. Adaptive information filtering: Learning in the presence of concept drifts. *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 33–40, 1998.

[134] Ron Kohavi, David H Wolpert, et al. Bias plus variance decomposition for zero-one loss functions. In *ICML*, volume 96, pages 275–83, 1996.

[135] J Zico Kolter and Marcus A Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8(Dec):2755–2790, 2007.

[136] Michał Koziarski, Bartosz Krawczyk, and Michał Woźniak. Radial-based oversampling for noisy imbalanced data classification. *Neurocomputing*, 343:19–33, 2019.

[137] 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.

[138] Michał Koziarski. CSMOUTE: Combined synthetic oversampling and undersampling technique for imbalanced data classification, 2021.

[139] B. Krawczyk, M. Koziarski, and M. Woźniak. Radial-based oversampling for multiclass imbalanced data classification. *IEEE Transactions on Neural Networks and Learning Systems*, 31(8):2818–2831, 2020.

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

[141] Bartosz Krawczyk and Alberto Cano. Online ensemble learning with abstaining classifiers for drifting and noisy data streams. *Applied Soft Computing*, 68:677–692, 2018.

[142] Bartosz Krawczyk and Bridget T McInnes. Local ensemble learning from imbalanced and noisy data for word sense disambiguation. *Pattern Recognition*, 78:103–119, 2018.

[143] Bartosz Krawczyk, Leandro L Minku, João Gama, Jerzy Stefanowski, and Michał Woźniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156, 2017.

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

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

[146] Georg Krempl and Vera Hofer. Classification in presence of drift and latency. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 596–603. IEEE, 2011.

[147] Georg Krempl, Indre Žliobaite, Dariusz Brzeziński, Eyke Hüllermeier, Mark Last, Vincent Lemaire, Tino Noack, Ammar Shaker, Sonja Sievi, Myra Spiliopoulou, et al. Open challenges for data stream mining research. *ACM SIGKDD Explorations Newsletter*, 16(1):1–10, 2014.

[148] Anders Krogh and Peter Sollich. Statistical mechanics of ensemble learning. *Physical Review E*, 55(1):811, 1997.

[149] Pawel Ksieniewicz. Standard decision boundary in a support-domain of fuzzy classifier prediction for the task of imbalanced data classification. In *International Conference on Computational Science*, pages 103–116. Springer, 2020.

[150] Paweł Ksieniewicz and Paweł Zyblewski. stream-learn–open-source python library for difficult data stream batch analysis. *arXiv preprint arXiv:2001.11077*, 2020.

[151] Wojciech Książek, Mohamed Hammad, Paweł Pławiak, U. Rajendra Acharya, and Ryszard Tadeusiewicz. Development of novel ensemble model using stacking learning and evolutionary computation techniques for automated hepatocellular carcinoma detection. *Biocybernetics and Biomedical Engineering*, 40(4):1512–1524, 2020.

[152] Miroslav Kubat and Stan Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In *In Proceedings of the Fourteenth International Conference on Machine Learning*, pages 179–186. Morgan Kaufmann, 1997.

[153] L.I. Kuncheva and S.T. Hadjitodorov. Using diversity in cluster ensembles. In *2004 IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1214–1219 vol.2, 2004.

[154] Ludmila I Kuncheva. Classifier ensembles for changing environments. In *International Workshop on Multiple Classifier Systems*, pages 1–15. Springer, 2004.

[155] Ludmila I. Kuncheva, James C. Bezdek, and Robert P.W. Duin. Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition*, 34(2):299–314, 2001.

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

[157] Balaji Lakshminarayanan, Daniel M Roy, and Yee Whye Teh. Mondrian forests: efficient online random forests. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pages 3140–3148, 2014.

[158] Doug Laney. 3d data management: Controlling data volume, velocity and variety. *META group research note*, 6(70):1, 2001.

[159] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[160] Joffrey L Leevy, Taghi M Khoshgoftaar, Richard A Bauder, and Naeem Seliya. A survey on addressing high-class imbalance in big data. *Journal of Big Data*, 5(1):42, 2018.

[161] Stanisław Lem. *The Astronauts*. Czytelnik, Poland, 1951.

[162] Stanisław Lem. *The Magellanic Cloud*. Iskry, Poland, 1955.

[163] Stanisław Lem. *Mortal Engines*. Czytelnik, Poland, 1964.

[164] Stanisław Lem. *Return from the Stars*. A Harvest/HBJ Book. Harcourt Brace Jovanovich, 1989.

[165] Peter J. Rousseeuw Leonard Kaufman. *Finding groups in data: An introduction to cluster analysis*. Wiley-Interscience, 1990.

[166] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1885–1893. Curran Associates, Inc., 2016.

[167] Zeng Li, Wenchao Huang, Yan Xiong, Siqi Ren, and Tuanfei Zhu. Incremental learning imbalanced data streams with concept drift: The dynamic updated ensemble algorithm. *Knowledge-Based Systems*, 195:105694, 2020.

[168] Nan-Ying Liang, Guang-Bin Huang, Paramasivan Saratchandran, and Narasimhan Sundararajan. A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, 17(6):1411–1423, 2006.

[169] Thomas Liebig, Nico Piatkowski, Christian Bockermann, and Katharina Morik. Dynamic route planning with real-time traffic predictions. *Information Systems*, 64:258–265, 2017.

[170] Márcia Lima, Victor Valle, Estevão Costa, Fylype Lira, and Bruno Gadelha. Software engineering repositories: Expanding the promise database. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, pages 427–436. ACM, 2019.

[171] Charles X Ling. Overfitting and generalization in learning discrete patterns. *Neurocomputing*, 8(3):341–347, 1995.

[172] Jiachen Liu, Qiguang Miao, Yanan Sun, Jianfeng Song, and Yining Quan. Modular ensembles for one-class classification based on density analysis. *Neurocomputing*, 171:262–276, 2016.

[173] Shigang Liu, Yu Wang, Jun Zhang, Chao Chen, and Yang Xiang. Addressing the class imbalance problem in twitter spam detection using ensemble learning. *Computers & Security*, 69:35–49, 2017.

[174] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2009.

[175] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2018.

[176] Menghua Luo, Ke Wang, Zhiping Cai, Anfeng Liu, Yangyang Li, and Chak Fong Cheang. Using imbalanced triangle synthetic data for machine learning anomaly detection. *Computers, Materials and Continua*, 58(1):15–26, 2019.

[177] Zhuo Ma, Yang Liu, Ximeng Liu, Jianfeng Ma, and Kui Ren. Lightweight privacy-preserving ensemble classification for face recognition. *IEEE Internet of Things Journal*, 6(3):5778–5790, 2019.

[178] Tomasz Maciejewski and Jerzy Stefanowski. Local neighbourhood extension of SMOTE for mining imbalanced data. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011, part of the IEEE Symposium Series on Computational Intelligence 2011, April 11-15, 2011, Paris, France*, pages 104–111, 2011.

[179] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5-th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[180] Gary R Marrs, Ray J Hickey, and Michaela M Black. The impact of latency on online classification learning with concept drift. In *International Conference on Knowledge Science, Engineering and Management*, pages 459–469. Springer, 2010.

[181] Colin McDiarmid et al. On the method of bounded differences. *Surveys in Combinatorics*, 141(1):148–188, 1989.

[182] Donald Michie. Methodologies from machine learning in data analysis and software. *The Computer Journal*, 34(6):559–565, 1991.

[183] Leandro L. Minku, Allan P. White, and Xin Yao. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 22(5):730–742, 2010.

[184] Leandro L. Minku and Xin Yao. DDD: A new ensemble approach for dealing with concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 24(4):619–633, 2012.

[185] Bilal Mirza and Zhiping Lin. Meta-cognitive online sequential extreme learning machine for imbalanced and concept-drifting data classification. *Neural Networks*, 80:79–94, 2016.

[186] Bilal Mirza, Zhiping Lin, and Nan Liu. Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift. *Neurocomputing*, 149:316–329, 2015.

[187] Bilal Mirza, Zhiping Lin, and Kar-Ann Toh. Weighted online sequential extreme learning machine for class imbalance learning. *Neural Processing Letters*, 38(3):465–486, 2013.

[188] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1 edition, 1997.

[189] Sankha Subhra Mullick, Shounak Datta, Sourish Gunesh Dhekane, and Swagatam Das. Appropriateness of performance indices for imbalanced data classification: An analysis. *Pattern Recognition*, 102:107197, 2020.

[190] K. Napierala and J. Stefanowski. Identification of different types of minority class examples in imbalanced data. In *Hybrid Artificial Intelligent Systems*, volume 7209 of *Lecture Notes in Computer Science*, pages 139–150. Springer Berlin Heidelberg, 2012.

[191] Krystyna Napierala and Jerzy Stefanowski. Types of minority class examples and their influence on learning classifiers from imbalanced data. *Journal of Intelligent Information Systems*, 46(3):563–597, 2016.

[192] Krystyna Napierała, Jerzy Stefanowski, and Izabela Szczech. Increasing the interpretability of rules induced from imbalanced data by using bayesian confirmation measures. In *International Workshop on New Frontiers in Mining Complex Patterns*, pages 84–98. Springer, 2016.

[193] Wing W. Y. Ng, Junjie Hu, Daniel S. Yeung, Shaohua Yin, and Fabio Roli. Diversified sensitivity-based undersampling for imbalance classification problems. *IEEE Transactions on Cybernetics*, 45(11):2402–2412, 2015.

[194] Kyosuke Nishida and Koichiro Yamauchi. Detecting concept drift using statistical testing. In *Discovery Science*, pages 264–269, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[195] Nikunj C. Oza and Stuart J. Russell. Online bagging and boosting. In Thomas S. Richardson and Tommi S. Jaakkola, editors, *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics*, volume R3 of *Proceedings of Machine Learning Research*, pages 229–236. PMLR, 04–07 Jan 2001.

[196] Malay K Pakhira. A linear time-complexity k-means algorithm using cluster shifting. In *2014 International Conference on Computational Intelligence and Communication Networks*, pages 1047–1051. IEEE, 2014.

[197] Shaoning Pang, Lei Zhu, Gang Chen, Abdolhossein Sarrafzadeh, Tao Ban, and Daisuke Inoue. Dynamic class imbalance learning for incremental lpsvm. *Neural Networks*, 44:87–100, 2013.

[198] D. Partridge and W. Krzanowski. Software diversity: practical statistics for its measurement and exploitation. *Information and Software Technology*, 39(10):707–717, 1997.

[199] Ripon Patgiri and Arif Ahmed. Big data: The v's of the game changer paradigm. In *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 17–24. IEEE, 2016.

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

[201] Ali Pesaranghader and Herna L. Viktor. Fast hoeffding drift detection method for evolving data streams. In *Machine Learning and Knowledge Discovery in Databases*, pages 96–111, Cham, 2016. Springer International Publishing.

[202] Ali Pesaranghader, Herna L Viktor, and Eric Paquet. Mcdiarmid drift detection methods for evolving data streams. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2018.

[203] Roberto HW Pinheiro, George DC Cavalcanti, and Ren Tsang. Combining binary classifiers in different dichotomy spaces for text categorization. *Applied Soft Computing*, 76:564–574, 2019.

[204] R. Polikar, L. Upda, S.S. Upda, and V. Honavar. Learn++: an incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 31(4):497–508, 2001.

[205] Arnu Pretorius, Surette Bierman, and Sarel J. Steel. A bias-variance analysis of ensemble learning for classification. *Annual Proceedings of the South African Statistical Association Conference*, 2016(1):57–64, 2016.

[206] S Priya and R Annie Uthra. Comprehensive analysis for class imbalance data with concept drift using ensemble based classification. *Journal of Ambient Intelligence and Humanized Computing*, 12(5):4943–4956, 2021.

[207] J Ross Quinlan. C4.5: Programs for machine learning. *The Morgan Kaufmann Series in Machine Learning*, 1993.

[208] Sergio Ramírez-Gallego, Bartosz Krawczyk, Salvador García, Michał Woźniak, and Francisco Herrera. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, 239:39–57, 2017.

[209] Siqi Ren, Bo Liao, Wen Zhu, and Keqin Li. Knowledge-maximized ensemble algorithm for different types of concept drift. *Information Sciences*, 430:261–281, 2018.

[210] Siqi Ren, Bo Liao, Wen Zhu, Zeng Li, Wei Liu, and Keqin Li. The gradual resampling ensemble for mining imbalanced data streams with concept drift. *Neurocomputing*, 286:150–166, 2018.

[211] Siqi Ren, Wen Zhu, Bo Liao, Zeng Li, Peng Wang, Keqin Li, Min Chen, and Zejun Li. Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning. *Knowledge-Based Systems*, 163:705–722, 2019.

[212] Andrew G. Barto Richard S. Sutton. *Reinforcement learning: an introduction*. Adaptive Computation and Machine Learning. The MIT Press, 1998.

[213] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, volume 3, pages 41–46, 2001.

[214] SW Roberts. Control chart tests based on geometric moving averages. *Technometrics*, 42(1):97–101, 2000.

[215] Frank Rosenblatt. Perceptron simulation experiments. *Proceedings of the IRE*, 48(3):301–309, 1960.

[216] Gordon J. Ross, Niall M. Adams, Dimitris K. Tasoulis, and David J. Hand. Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters*, 33(2):191–198, 2012.

[217] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[218] Neelam Rout, Debahuti Mishra, and Manas Kumar Mallick. Handling imbalanced data: A survey. In *International Proceedings on Advances in Soft Computing, Intelligent Systems and Applications*, pages 431–443. Springer, 2018.

[219] Daniel M Roy and Yee Whye Teh. The mondrian process. In *Proceedings of the 21st International Conference on Neural Information Processing Systems*, pages 1377–1384, 2008.

[220] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4393–4402. PMLR, 10–15 Jul 2018.

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

[222] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[223] Cullen Schaffer. Overfitting avoidance as bias. *Machine learning*, 10(2):153–178, 1993.

[224] Robert E. Schapire. The strength of weak learnability. *Mach. Learn.*, 5(2):197–227, July 1990.

[225] Bernhard Schölkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 582–588, Cambridge, MA, USA, 1999. MIT Press.

[226] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000.

[227] Clayton Scott, Gilles Blanchard, and Gregory Handy. Classification with asymmetric label noise: Consistency and maximal denoising. In *Conference On Learning Theory*, pages 489–511, 2013.

[228] Burr Settles. Active learning. In *Synthesis Lectures on Artificial Intelligence and Machine Learning*, volume 6, pages 1–114. Morgan & Claypool Publishers, 2012.

[229] Martin Sewell. Machine learning, 2006.

[230] Ammar Shaker and Eyke Hüllermeier. Recovery analysis for adaptive learning from non-stationary data streams: Experimental design and case study. *Neurocomputing*, 150:250–264, 2015.

[231] Shiven Sharma, Colin Bellinger, Bartosz Krawczyk, Osmar Zaiane, and Nathalie Japkowicz. Synthetic oversampling with the majority class: A new perspective on handling extreme imbalance. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 447–456. IEEE, 2018.

[232] David B Skalak et al. The sources of increased accuracy for two proposed boosting algorithms. In *Proc. American Association for Artificial Intelligence, AAAI-96, Integrating Multiple Learned Models Workshop*, volume 1129, page 1133, 1996.

[233] Michael R Smith, Tony Martinez, and Christophe Giraud-Carrier. An instance level analysis of data complexity. *Machine Learning*, 95(2):225–256, 2014.

[234] Padhraic Smyth and David Wolpert. Stacked density estimation. In *Proceedings of the 10th International Conference on Neural Information Processing Systems*, pages 668–674, 1997.

[235] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc: A family of discriminant measures for performance evaluation. In *AI 2006: Advances in Artificial Intelligence*, pages 1015–1021, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

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

[237] W. Nick Street and YongSeog Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 377–382, New York, NY, USA, 2001. ACM.

[238] Jiang Su and Harry Zhang. A fast decision tree learning algorithm. In *AAAI*, volume 6, pages 500–505, 2006.

[239] Xiao-Yan Sun, Hua-Xiang Zhang, and Zhi-Chao Wang. Clustering based bagging algorithm on imbalanced data sets. In *International Symposium on Integrated Uncertainty in Knowledge Modelling and Decision Making*, pages 179–186. Springer, 2011.

[240] Yange Sun, Yi Sun, and Honghua Dai. Two-stage cost-sensitive learning for data streams with concept drift and class imbalance. *IEEE Access*, 8:191942–191955, 2020.

[241] Yu Sun, Ke Tang, Leandro L Minku, Shuo Wang, and Xin Yao. Online ensemble learning of data streams with gradually evolved classes. *IEEE Transactions on Knowledge and Data Engineering*, 28(6):1532–1545, 2016.

[242] Ryszard Tadeusiewicz. Introduction to intelligent systems. In *Intelligent systems*, pages 1–1. CRC Press, 2018.

[243] Muhammad Atif Tahir, Josef Kittler, and Ahmed Bouridane. Multilabel classification using heterogeneous ensemble of multi-label classifiers. *Pattern Recognition Letters*, 33(5):513–523, 2012.

[244] Yuchun Tang, Yan-Qing Zhang, Nitesh V Chawla, and Sven Krasser. Svms modeling for highly imbalanced classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(1):281–288, 2008.

[245] David MJ Tax and Robert PW Duin. Support vector domain description. *Pattern Recognition Letters*, 20(11-13):1191–1199, 1999.

[246] David MJ Tax and Robert PW Duin. Combining one-class classifiers. In *International Workshop on Multiple Classifier Systems*, pages 299–308. Springer, 2001.

[247] David MJ Tax and Robert PW Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.

[248] Fadi Thabtah, Suhel Hammoud, Firuz Kamalov, and Amanda Gonsalves. Data imbalance in classification: Experimental evaluation. *Information Sciences*, 513:429–441, 2020.

[249] Kai Ming Ting. A comparative study of cost-sensitive boosting algorithms. In *In Proceedings of the 17th International Conference on Machine Learning*. Citeseer, 2000.

[250] Mariusz Topolski. Application of the stochastic gradient method in the construction of the main components of pca in the task diagnosis of multiple sclerosis in children. In *International Conference on Computational Science*, pages 35–44. Springer, 2020.

[251] IBA Turing. Computing machinery and intelligence-am turing. *Mind*, 59(236):433, 1950.

[252] Vimal B Vaghela, Amit Ganatra, and Amit Thakkar. Boost a weak learner to a strong learner using ensemble system approach. In *2009 IEEE International Advance Computing Conference*, pages 1432–1436. IEEE, 2009.

[253] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984.

[254] Carlos Valle, Francisco Saravia, Héctor Allende, Raúl Monge, and César Fernández. Parallel approach for ensemble learning with locally coupled neural networks. *Neural Processing Letters*, 32(3):277–291, 2010.

[255] Jason D Van Hulse, Taghi M Khoshgoftaar, and Haiying Huang. The pairwise attribute noise detection algorithm. *Knowledge and Information Systems*, 11(2):171–190, 2007.

[256] Vladimir N Vapnik. The nature of statistical learning. *Theory*, 1995.

[257] Michel Verleysen and Damien François. The curse of dimensionality in data mining and time series prediction. In *Computational Intelligence and Bioinspired Systems*, pages 758–770, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[258] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, page 226–235, New York, NY, USA, 2003. Association for Computing Machinery.

[259] Qiyue Wang, Yao Lu, Xiaoke Zhang, and James Hahn. Region of interest selection for functional features. *Neurocomputing*, 422:235–244, 2021.

[260] S WANG, LL MINKU, and X YAO. Dealing with multiple classes in online class imbalance learning. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2118–2124. AAAI Press, 2016.

[261] Shuo Wang, Leandro L Minku, and Xin Yao. A learning framework for online class imbalance learning. In *2013 IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL)*, pages 36–45. IEEE, 2013.

[262] Shuo Wang, Leandro L Minku, and Xin Yao. Resampling-based ensemble methods for online class imbalance learning. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1356–1368, 2014.

[263] Shuo Wang and Xin Yao. Multiclass imbalance problems: Analysis and potential solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(4):1119–1130, 2012.

[264] Yi Wang, Yang Zhang, and Yong Wang. Mining data streams with skewed distribution by static classifier ensemble. In *Opportunities and Challenges for Next-Generation Applied Intelligence*, pages 65–71. Springer, 2009.

[265] Zeyuan Wang, Josiah Poon, and Simon Poon. Ami-net+: A novel multi-instance neural network for medical diagnosis from incomplete and imbalanced data. *arXiv preprint arXiv:1907.01734*, 2019.

[266] Marco Wiering and Martijn Van Otterlo. Reinforcement learning. *Adaptation, Learning, and Optimization*, 12, 2012.

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

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

[269] Michał Woźniak. Application of combined classifiers to data stream classification. In *IFIP International Conference on Computer Information Systems and Industrial Management*, pages 13–23. Springer, 2013.

[270] Michał Woźniak. *Hybrid Classifiers: Methods of Data, Knowledge, and Classifier Combination*. Springer Berlin Heidelberg, 2014.

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

[272] Qiang Wu and Ding-Xuan Zhou. Analysis of support vector machine classification. *Journal of Computational Analysis & Applications*, 8(2), 2006.

[273] Xindong Wu, Peipei Li, and Xuegang Hu. Learning from concept drifting data streams with unlabeled data. *Neurocomputing*, 92:145–155, 2012.

[274] Xindong Wu, Xingquan Zhu, Gong-Qing Wu, and Wei Ding. Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):97–107, 2014.

[275] Yingchao Xiao, Huangang Wang, Lin Zhang, and Wenli Xu. Two methods of selecting gaussian kernel parameters for one-class svm and their application to fault detection. *Knowledge-Based Systems*, 59:75–84, 2014.

[276] XU Xiaolong, CHEN Wen, and SUN Yanfei. Over-sampling algorithm for imbalanced data classification. *Journal of Systems Engineering and Electronics*, 30(6):1182–1191, 2019.

[277] Hong-Jie Xing, Ya-Jie Liu, and Zi-Chuan He. Robust sparse coding for one-class classification based on correntropy and logarithmic penalty function. *Pattern Recognition*, page 107685, 2020.

[278] L. Xu, A. Krzyzak, and C.Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):418–435, 1992.

[279] Shuliang Xu and Junhong Wang. Dynamic extreme learning machine for data stream classification. *Neurocomputing*, 238:433–449, 2017.

[280] Y. Xu. Maximum margin of twin spheres support vector machine for imbalanced data classification. *IEEE Transactions on Cybernetics*, 47(6):1540–1550, 2017.

[281] Ying Yang, Xindong Wu, and Xingquan Zhu. Dealing with predictive-but-unpredictable attributes in noisy data sources. In *Knowledge Discovery in Databases: PKDD 2004*, pages 471–483. Springer Berlin Heidelberg, 2004.

[282] Xue Ying. An overview of overfitting and its solutions. In *Journal of Physics: Conference Series*, volume 1168. IOP Publishing, 2019.

[283] Hualong Yu and Geoffrey I Webb. Adaptive online extreme learning machine by regulating forgetting factor by concept drift map. *Neurocomputing*, 343:141–153, 2019.

[284] Lean Yu, Xiaowen Huang, and Hang Yin. Can machine learning paradigm improve attribute noise problem in credit risk classification? *International Review of Economics & Finance*, 70:440–455, 2020.

[285] X. Yuan, P. He, Q. Zhu, and X. Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9):2805–2824, 2019.

[286] Cha Zhang and Yunqian Ma. *Ensemble machine learning: methods and applications*. Springer-Verlag New York, 2012.

[287] Changqing Zhang, Ziwei Yu, Huazhu Fu, Pengfei Zhu, Lei Chen, and Qinghua Hu. Hybrid noise-oriented multilabel learning. *IEEE Transactions on Cybernetics*, 50(6):2837–2850, 2020.

[288] Yan-Ping Zhang, Li-Na Zhang, and Yong-Cheng Wang. Cluster-based majority under-sampling approaches for class imbalance learning. In *2010 2nd IEEE International Conference on Information and Financial Engineering*, pages 400–404. IEEE, 2010.

[289] Yuyan Zhang, Xinyu Li, Liang Gao, Lihui Wang, and Long Wen. Imbalanced data fault diagnosis of rotating machinery using synthetic oversampling and feature learning. *Journal of Manufacturing Systems*, 48:34–50, 2018.

[290] Peilin Zhao, Yifan Zhang, Min Wu, Steven CH Hoi, Mingkui Tan, and Junzhou Huang. Adaptive cost-sensitive online classification. *IEEE Transactions on Knowledge and Data Engineering*, 31(2):214–228, 2018.

[291] Qiang Li Zhao, Yan Huang Jiang, and Ming Xu. Incremental learning by heterogeneous bagging ensemble. In *International Conference on Advanced Data Mining and Applications*, pages 1–12. Springer, 2010.

[292] Yong-Ping Zhao, Gong Huang, Qian-Kun Hu, and Bing Li. An improved weighted one class support vector machine for turboshaft engine fault detection. *Engineering Applications of Artificial Intelligence*, 94:103796, 2020.

[293] Xueyuan Zhou and Mikhail Belkin. Semi-supervised learning. In *Academic Press Library in Signal Processing*, volume 1, pages 1239–1269. Elsevier, 2014.

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

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

[296] Fa Zhu, Jian Yang, Cong Gao, Sheng Xu, Ning Ye, and Tongming Yin. A weighted one-class support vector machine. *Neurocomputing*, 189:1–10, 2016.

[297] Xiaojin Zhu and Andrew Goldberg. *Introduction to Semi-Supervised Learning*. Morgan & Claypool Publishers, 2009.

[298] Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review*, 22(3):177–210, 2004.

[299] Xingquan Zhu, Xindong Wu, and Qijun Chen. Eliminating class noise in large datasets. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pages 920–927. AAAI Press, 2003.

[300] Yujin Zhu, Zhe Wang, and Daqi Gao. Gravitational fixed radius nearest neighbor for imbalanced problem. *Knowledge-Based Systems*, 90:224–238, 2015.

[301] Paweł Zyblewski, Paweł Ksieniewicz, and Michał Woźniak. Classifier selection for highly imbalanced data streams with minority driven ensemble. In *International Conference on Artificial Intelligence and Soft Computing*, pages 626–635. Springer, 2019.