Wydział Elektroniki
Katedra Systemów i Sieci Komputerowych
Politechnika Wrocławska

Classifier selection for imbalanced data stream classification

Selekcja klasyfikatorów w zadaniu klasyfikacji niezbalansowanych strumieni danych

Seria: PRE nr W04/2021/P-009

Paweł Zyblewski

Słowa kluczowe: *Rozpoznawanie wzorców; uczenie indukcyjne; klasyfikacja; zespół klasyfikatorów; selekcja klasyfikatorów; przetwarzanie wstępne danych; dane trudne; dane niezbalansowane; strumienie danych; dryf koncepcji; uczenie aktywne.*

Wrocław 04.2021

WROCLAW UNIVERSITY OF SCIENCE AND TECHNOLOGY

Classifier selection for imbalanced data stream classification

by

Pawel Zyblewski

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

in the
Faculty of Electronics
Department of Systems and Computer Networks

April 2021

With the sun in my hand
Gonna throw the sun
Way across the land-
Cause I'm tired,
Tired as I can be

So Tired Blues by Langston Hughes

Acknowledgements

First and foremost, I would like to express my most sincere gratitude to my supervisor, Prof. Michał Woźniak. He is the person who gave me the opportunity to start a scientific career and thus find a passion that – to my surprise – is supposedly perceived by some as a job.

I am very grateful to Dr. Paweł Ksieniewicz, who, for all intents and purposes, acted as my co-supervisor. From him I learned how to write scientific code, what it means to be busy, and that typography can be quite interesting.

I also would like to show appreciation to Prof. Robert Sabourin from École de Technologie Supérieure in Montreal, Canada, for the collaboration that formed the basis for part of this thesis.

I give thanks to my colleagues from the Department of Systems and Computer Networks: Jakub Klikowski, Szymon Wojciechowski, Dr. Wojciech Kmiecik, Dr. Paweł Trajdos, Dr. Dariusz Jankowski, and others for the friendly atmosphere and for making working at the university a pleasure. I would also like to express my gratitude to Dominika Sułot from the Department of Biomedical Engineering for supporting me in my decision to start a PhD studies and for our joint research work.

And last but not least, I would like to thank all my friends for putting up with my constant babbling about the university, always supporting me (in different areas of life) and creating a family that allows me to detach from my studies.

Abbreviations

AUC	**A**rea **U**nder **C**urve
BAC	**B**alanced **A**ccuracy
BALS	**B**udget **A**ctive **L**abeling **S**trategy
DEP	**D**iversity **E**nsemble **P**runing
DESISC	**D**ynamic **E**nsemble **S**election for **I**mbalanced **S**tream **C**lassification
DESISC-SB	**DESISC** using **S**tratified **B**agging
FN	**F**alse **N**egative – number of wrong classified positive examples
FNR	**F**alse **N**egative **R**ate (*miss rate*)
FP	**F**alse **P**ositive – number of wrong classified negative examples
FPR	**F**alse **P**ositive **R**ate (*fallout*)
KNORA	**k**-**N**earest **O**racle
MDE	**M**inority **D**riven **E**nsemble
RSMO	**R**andom **S**ampling **M**ultistage **O**rganization
SEA	**S**treaming **E**nsemble **A**lgorithm
TN	**F**rue **N**egative – number of correct classified negative examples
TNR	**T**rue **N**egative **R**ate (*specificity*)
TP	**T**rue **P**ositive – number of correct classified positive examples
TPR	**T**rue **P**ositive Rate (*recall, sensitivity*)
TSMV	**T**wo-**S**tep **M**ajority **V**oting

Symbols

d	feature vector dimension
c	number of clusters
e	number of iterations/epochs
D_j	decision area for class j
DS	data stream
DS_i	ith chunk of data stream DS
$\mathcal{DSEL}$	dynamic selection dataset
E	number of pairwise overlaps between a set of spherical classifiers
f	probability density function
f_j	conditional probability density function
F	support function
F_i	support function for class i
F_i^k	support function of kth classifier for class i
$\mathcal{F}$	set of support functions
$Gmean$	geometric mean of precision and recall
$Gmean_s$	geometric mean of specificity
i, j	class label
$\mathbf{J}$	random variable
L	loss function

$\mathcal{LS}$	learning set
$\mathcal{LS}_i$	subset of learning set which includes examples from the i-th class only
$\mathcal{M}$	set of class labels
P_acc	probability of accuracy
P_err	probability of error
p_i	a prior probability of the i-th class
$p_i(x)$	a posterior probability of x given i
Π	pool of base classifiers
n	number of base classifiers
Ψ	classification algorithm
Ψ^*	optimal (Bayes) classifier
$Risk(\Psi)$	average (overall) risk of classifier Ψ
$\mathcal{TS}$	testing set
$\mathcal{VS}$	validation set
N	cardinality of a set
x	feature vector
$x^{(l)}$	the lth feature
x_i^k	kth example in ith data chunk
$\mathcal{X}$	feature space
X_i	the i-th constituent of $\mathcal{X}$
$\mathbf{X}$	random variable
$[\,]$	Iverson bracket $[true] = 1$, $[false] = 0$

Abstract

The thesis focuses on the use of the *Dynamic Ensemble Selection* algorithms in conjunction with data preprocessing techniques in the tasks of the stream and imbalanced data classification. The aim was to present the natural ability of classifier selection algorithms to deal with data imbalance and to propose new, effective solutions to the rarely discussed problem of highly imbalanced data stream classification. Based on these assumptions, the following hypothesis was formulated

> *There exist such methods employing data preprocessing and classifier selection that can outperform state-of-the-art classifiers for difficult data classification tasks.*

The hypothesis was substantiated by achieving the following goals:
Goal 1 – **Developing an ensemble selection algorithm for imbalanced data classification, as well as designing a dedicated combination rule.**
This goal was met by developing three algorithms based on the clustering of models in a one-dimensional space of classifier diversity. To construct this clustering space, the H measure, informing about about the impact of individual classifiers on the ensemble diversity, was proposed.

The *Diversity Ensemble Pruning* (DEP) prunes the ensemble by selecting, from each cluster, only the model with the highest BAC value. The *Two-step majority voting organization* (TSMV) algorithm classifies imbalanced data using the two-step voting structure. The *Random Sampling Multistage Organization* (RSMO) algorithm, additionally uses sampling with replacement to reduce the number of similar models involved in the decision-making process.

Goal 2 – **Proposing a novel distance-based *Dynamic Ensemble Selection* method for imbalanced data classification.**
This goal was met by proposing novel *Dynamic Classifier Selection* algorithms for the imbalanced data classification problem. Two methods were proposed, namely *Dynamic*

Ensemble Selection using Euclidean distance (DESE) and *Dynamic Ensemble Selection using Imbalance Ratio and Euclidean distance* (DESIRE), which use the Euclidean distance and *Imbalance Ratio* in the training set to select the most appropriate model for the classification of each new sample. DESE performs the selection based on local competencies and distance to classified neighbors, while DESIRE additionally scales the obtained weights by *Imbalance Ratio* of the problem.

Goal 3 – **Developing a chunk-based ensemble algorithm, aimed specifically for the task of highly imbalanced data stream classification.**
This goal was achieved by proposing the *Minority Driven Ensemble* (MDE) algorithm. This algorithm classifies highly imbalanced data streams using a decision rule exploiting local data characteristics to prefer the minority class instances.

Goal 4 – **Designing a novel framework combining *Dynamic Ensemble Selection* and preprocessing techniques for imbalanced data stream classification.**
This goal was achieved by proposing two batch-based approaches, combining *Dynamic Classifier Selection* algorithms and preprocessing techniques for the task of highly imbalanced data stream classification. The *Dynamic Ensemble Selection for Imbalanced Stream Classification* (DESISC) method generates a single model on each data chunk, while the *Dynamic Ensemble Selection for Imbalanced Stream Classification approach using Stratified Bagging* (DESISC-SB) employs a stratified version of *Bagging* for the base classifier generation.

Goal 5 – **Proposing a strategy for learning from drifting data stream under limited access to labels scenario.**
This goal was achieved by the introduction of the *Budget Active Labeling Strategy* (BALS) algorithm. The proposed approach, in addition to the pool of objects selected for labeling based on their distance to the decision boundary, also received a small number of randomly selected objects.

Goal 6 – **Evaluating the behavior of the previously proposed data stream classification framework, taking into account the limitation in the label access.**
This goal was achieved by combining the proposed DESISC-SB framework with the active learning method based on selecting patterns located at a certain distance from the decision boundary.

Goal 7 – **Conducting an experimental evaluation of the proposed methods in comparison to *state-of-the-art* approaches.**
Goal 8 – **Developing a *Python* Machine Learning library for difficult data stream analysis.**

Goals 7 and 8 were achieved by designing an experimental environment for imbalanced data classification, as well as by creating the *stream-learn*[1] package for difficult data stream analysis, which was used to conduct all experiments related to data stream classification.

Keywords
Pattern recognition; inductive learning; classification; classifier ensemble; classifier selection; difficult data; imbalanced data; data stream; data preprocessing; concept drift; active learning.

[1]Ksieniewicz, P. and Zyblewski, P., 2020. stream-learn–open-source Python library for difficult data stream batch analysis. arXiv preprint arXiv:2001.11077.

Streszczenie

Rozprawa doktorska koncentruje się na wykorzystaniu algorytmów *Dynamicznej Selekcji Zespołu Klasyfikatorów* w połączeniu z metodami przetwarzania wstępnego w zadaniu klasyfikacji statycznych oraz strumieniowych danych niezbalansowanych. Celem pracy było przedstawione naturalnej zdolności algorytmów selekcji klasyfikatorów do radzenia sobie z niezbalansowaniem danych oraz zaproponowanie nowych, efektywnych rozwiązań rzadko poruszanego w literaturze problemu klasyfikacji wysoce niezbalansowanych strumieni danych. W oparciu o te założenia, w pracy sformułowa została hipoteza, zakładająca, że

Istnieją metody wykorzystujące zarówno wstępne przetwarzanie danych, jak i metody selekcji klasyfikatorów, które przewyższają jakość predykcji znanych z literatury metod stosowanych w klasyfikacji danych trudnych.

Hipoteza została uprawdopodobniona poprzez osiągnięcie poniższych celów:

Cel 1 **– Opracowanie algorytmu selekcji zespołu klasyfikatorów na potrzeby klasyfikacji danych niezbalansowanych oraz zaprojektowanie dedykowanej reguły kombinacji.**
Cel został zrealizowany poprzez opracowanie trzech algorytmów, opartych na grupowaniu modeli bazowych w jednowymiarowej przestrzeni różnorodności klasyfikatorów. Podstawę do utworzenia tej przestrzeni stanowiła zaproponowana miara H, informująca o wpływie poszczególnych klasyfikatorów na różnorodność osiąganą przez cały zespół.

Algorytm *Diversity Ensemble Pruning* (DEP) dokonuje grupowania modeli bazowych w przestrzeni różnorodności, a następnie ocenia jakość klasyfikacji poszczególnych klasyfikatorów w oparciu o *zbalansowaną dokładność.* Do finalnego zespołu wybierany jest, z każdego klastra, model o najwyższej wartości BAC. Algorytm *Two-step majority voting organization* (TSMV), zamiast redukować liczność zespołu, dokonuje klasyfikacji danych niezbalansowanych z wykorzystaniem struktury głosowania dwuetapowego. W pierwszym etapie głosowania, każdy klaster traktowany jest jako osobny zespół klasyfikatorów, który niezależnie podejmuje decyzję w oparciu o *głosowanie większościowe.*

W drugim etapie, ponownie poprzez *głosowanie większościowe*, kombinowane są decyzje uzyskane przez poszczególne klastry. Algorytm *Random Sampling Multistage Organization* (RSMO), będący modyfikacją TSVM, wykorzystuje dodatkowo operację losowania ze zwracaniem w celu zredukowanie liczby podobnych klasyfikatorów wykorzystywaych w procesie podejmowania decyzji.

Cel 2 **– Opracowanie algorytmu *Dynamicznej Selekcji Klasyfikatorów* opartego o miary dystansu, na potrzeby klasyfikacji danych niezbalansowanych.**
Cel został zrealizowany poprzez opracowanie dwóch algorytmów Dynamicznej Selekcji Klasyfikatorów, które oceniają kompetencje modeli bazowych w zależności od decyzji podjętych przez nie w odniesieniu do przypadków znajdujących się w lokalnym sąsiedztwie klasyfikowanej instancji, jednocześnie uwzględniając odległość Euklidesową do tych przypadków. *Dynamic Ensemble Selection using Euclidean distance* (DESE) wykorzystuje do selekcji wyłącznie decyzje klasyfikatorów oraz odległości, natomiast *Dynamic Ensemble Selection using Imbalance Ratio and Euclidean distance* (DESIRE) dodatkowo modyfikuje otrzymane wagi w oparciu o stopień niezbalansowania klasyfikowanego problemu.

Cel 3 **– Opracowanie opartego o przetwarzanie wsadowe algorytmu klasyfikacji wysoce niezbalansowanych strumieni danych.**
Cel został zrealizowany poprzez zaproponowanie algorytmu *Minority Driven Ensemble* (MDE). Algorytm ten dokonuje klasyfikacji wysoce niezbalansowanych strumieni danych z użyciem reguły decyzyjnej, która wykorzystuje lokalną charakterystykę danych do preferowania klasy mniejszościowej.

Cel 4 **– Zaprojektowanie metody łączącej Dynamiczną Selekcją Klasyfikatorów oraz przetwarzanie wstępne danych, na potrzeby klasyfikacji niezbalansowanych danych strumieniowych.**
Cel został osiągnięty poprzez zaproponowanie dwóch, opartych o przetwarzanie wsadowe, podejść do łączenia algorytmów Dynamicznej Selekcji Klasyfikatorów oraz technik przetwarzania wstępnego na potrzeby klasyfikacji wysoce niezbalansowanych strumieni danych. Metoda *Dynamic Ensemble Selection for Imbalanced Stream Classification* (DESISC) generuje pojedynczy model na każdej nowej porcji danych, podczas gdy podejście *Dynamic Ensemble Selection for Imbalanced Stream Classification using Stratified Bagging* (DESISC-SB) wykorzystuje do tego celu stratyfikowaną wersję *Baggingu.*

Cel 5 **– Zaproponowanie strategii budowania modeli klasyfikacji w przypadku strumieni danych z ograniczonym dostępem do etykiet.**
Cel został osiągnięty poprzez zapoponowanie strategii odpytywania o etykiety, nazwanej *Budget Active Labeling Strategy* (BALS). Algorytm ten łączy w sobie losowe podejście do etykietyzacji z podejściem właściwym algorytmom uczenia aktywnego. Dzięki temu oprócz puli instancji wybranych na podstawie ich odległości od granicy decyzyjnej,

etykiety pozyskiwane są również dla małej liczby obiektów losowo wybranych z aktualnej porcji danych.

Cel 6 – **Ewaluacja zaproponowanego wcześniej frameworku klasyfikacji strumieni danych, w przypadku ograniczonego dostępu do etykiet.**
Cel został osiągnięty poprzez połączenie metody DESISC-SB z podejściem do uczenia aktywnego, opartym na przekazywaniu do etykietyzacji przypadków znajdujących się w określonej odległości od granicy decyzyjnej problemu.

Cel 7 – **Przeprowadzenie ewaluacji eksperymentalnej, porównującej zaproponowane algorytmy z podejściami stanowiącymi *state-of-the-art*.**
Cel 8 – **Opracowanie biblioteki języka *Python*, pozwalającej na analizę trudnych strumieni danych.**
Cele 7 i 8 zostały osiągnięte dzięki zaprojektowaniu oraz implementacji środowiska eksperymentalnego w języku *Python*, które posłużyło do przeprowadzenia badań związanych z klasyfikacją danych niezbalansowanych. Dodatkowo, w trakcie pracy na rozprawą, opracowana została biblioteka *stream-learn*[2], pozwalająca na przetwarzanie niezbalansowanych strumieni danych z dryfem koncepcji. Biblioteka ta została wykorzystana do przeprowadzenia wszystkich eksperymentów związanych z danymi strumieniowymi.

Słowa kluczowe
Rozpoznawanie wzorców; uczenie indukcyjne; klasyfikacja; zespół klasyfikatorów; selekcja klasyfikatorów; przetwarzanie wstępne danych; dane trudne; dane niezbalansowane; strumienie danych; dryf koncepcji; uczenie aktywne.

[2]Ksieniewicz, P. and Zyblewski, P., 2020. stream-learn–open-source Python library for difficult data stream batch analysis. arXiv preprint arXiv:2001.11077.

Contents

Chapter 1

Introduction

Nowadays, many practical classification tasks require building a model from data containing various serious difficulties. These complications may be represented by characteristics such as a high number of problem classes [6], data heterogeneity [186], the high dimensionality of the problem [216], low or very high cardinality of the learning set, or data incompleteness [247]. Regardless of which of these difficulties occurs in the analyzed data set, they can severely deteriorate the performance of the final model, and the problem containing at least one of them can be described as the task of difficult data classification. In the following thesis, data difficulty is defined mainly by the imbalanced class distributions [133] and streaming data [135].

The nature of imbalanced data and data stream

The primary problem with learning from imbalanced data is the ability of data with skewed class distribution to significantly deteriorate the performance of classical learning algorithms, as they assume a roughly equal number of samples in all considered problem classes [119]. However, in many real-life tasks, samples from some classes appear much more frequently than from others. In the case of binary classification, these classes are called majority and minority classes, respectively. Therefore, when confronted with the problem of imbalanced data classification, the above-mentioned algorithms fail to represent the data distributive characteristics and display a bias towards the majority class [133]. At the same time, from the point of view of the classification task, it is the minority class that is usually more important.

The problem of data stream classification is interesting due to a potentially infinite amount of continuously arriving data, which can appear at high speed and require a quick response from the decision system. Data streams pose new challenges for traditional machine learning algorithms, which were designed with the classification of static data in mind and are not capable of adapting to the characteristics exhibited by the fast growing amounts of data [135]. The most distinctive feature of a data stream is the phenomenon called *concept drift*, which can change the data distribution in the stream over time and thus lead to deterioration of the classification model. *Concept drift* can be categorized as (*i*) *virtual* or *real*, depending on the influence of the changes on the shape of the decision boundary, (*ii*) *sudden*, *gradual* or *incremental*, depending on the dynamics of changes, and (*iii*) *recurring* or *non-recurring*, depending on the possibility of the reappearance of previously observed concepts. Additional problems are memory and time constraints due to the potentially infinite amount of data as well as potential limitations in the ability to label all incoming samples.

The imbalanced data stream classification task [37], which combines both of the notions described above, is very rarely represented in the literature. This is despite the fact that real-life data streams often exhibit high and dynamically changing class imbalance. When dealing with both imbalanced data and data stream classification, one of the most promising directions is the approach based on classifier ensemble [150]. Ensemble methods, due to their flexibility, allow for easy combination with data preprocessing in the case of learning from imbalanced data and for the continuous adaptation of the classifier pool to deal with the concept drift occurrence. This approach refers to the need, rooted in human nature, to obtain a few opinions before making a decision. That is why the foundations of the need to generate relatively strong (better than random guess) and diverse (making mistakes on different instances of the problem [151]) models can also be found in political science. It is also worth paying attention to the important role of classifier selection [59], both static and dynamic, which allows for more effective use of the local knowledge of each base model.

Ensemble learning roots

Everyday decision making is an essential part of everyone's life. We think about trivial things. We decide what to eat for dinner, what to wear for work, or what book to read after coming back home. However, we also consider choices that have a much greater impact on our lives, such as choosing an education path, career, or buying a house. In many of these cases, we seek for a help in the opinion of an expert who has been gaining experience in a given field for years and – with given probability dependent to his or hers lifespan – is able to recommend us the best possible choice.

However, it is also worth considering an alternative that has long been considered by political science, namely the *Wisdom of Crowds* - initiated by Condorcet's jury theorem, which was first introduced by Marquis de Condorcet in 1785 in an important work on probability, *Essay on the Application of Analysis to the Probability of Majority Decisions* [48], which was originally published in French[1].

Condorcet's theorem provides the theoretical basis for democracy, describing the relative likelihood of a group of people reaching the right solution to a problem by combining their knowledge (by voting) and trusting the majority's decision. The conclusion being, that a majority of independent individuals who make correct decisions with a probability greater than by random choice are more likely to make the correct choice than each of the individual separately [212]. Unfortunately, the assumptions made by Condorcet were quite unrealistic and difficult to achieve in reality, however, there have also been some generalizations of this theory that no longer possess these limitations [153].

A well-known and often-cited example of the effectiveness of the *Wisdom of the Crowds* is the experiment carried out by the English statistician Francis Galton during a competition organized at the Plymouth fair. The aim of the competition was to guess the weight of the slaughtered and dressed ox, and the winner was the person whose proposal was closest to the real value. In his work *Vox Populi* [87], published in 1907, Galton described gathering 800 voting cards and – after getting rid of 13 unreadable ones – calculating the median of the remaining 787 votes in order to represent the combined wisdom of each participant. The result was a response of 1.207 pounds, which differed only by 1% from the true weight of 1.198 pounds. After the publication of *Vox Populi*, one of the readers started a discussion with Galton in which he proposed using the average of the votes instead of the median. It turned out that this approach led to a virtually perfect result, differing from the true value by only one pound.

Surowiecki, based on this phenomenon, concluded in his book *The wisdom of crowds* [226] that instead of looking for experts in a given field – which can often turn out to be a highly costly process – one should rather approach the crowd that may know the answer to the problem in question. He also referred to the show *Who Wants to Be a Millionaire* in which a player, if unsure about the question, can use one of the three lifelines. Two of these aids are, respectively, a phone call to a friend previously selected by the competitor, who may be considered an expert, and a request for the opinion of a random crowd located in a TV studio. According to the data provided by Surowiecki, the experts answered correctly almost 65 percent of the time, while the audience picked the right answer 91 percent of the time. Even without knowing the level of expert knowledge and the fact that these statistics do not relate to the same questions, there is a clear similarity between this example and the research conducted by Galton.

Based on this assumption as well as Condorcet's criterion, the desired properties of a decision-making system based on the group opinion of people can be listed [194]:

[1]*Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*

- Diversity of opinion – each member should possess different information and have a different perspective on the problem,
- Independence – each member should make independent decision errors [236],
- Decentralization – each member should draw conclusions based on their local knowledge,
- Aggregation – an approach to combining individual decisions into a joint result

The core of the ensemble learning approach and the reason for using it is perfectly rendered in the quote from Marvin Minsky's book *The Society of Mind* [173] - *"What magical trick makes us intelligent? The trick is that there is no trick. The power of intelligence stems from our vast diversity, not from any single, perfect principle."*

1.1 Motivation and challenges

The following thesis aims to connect two rarely combined research directions, i.e., non-stationary data stream classification and data analysis with skewed class distributions.

Learning from non-stationary data streams remains the focus of intense research because many real decision-making problems should process on streaming data [135]. Nevertheless, the decision-making algorithms should also take into consideration the disproportions among the observations from different classes [133]. Because real data streams may exhibit a high and changing class imbalance ratio, which can further hinder the classification task, then the high demand for this type of solution is evident.

A typical example of such a case is the technical diagnosis in which the fault probability increases with utilization time, and it may be a result of material fatigue. Sometimes the relationship between the minority and majority classes changes in a way that the former minority becomes the majority class. We may observe this phenomenon in tasks related to social media analysis, as the popularity of topics discussed on *Twitter* [223] or environmental hazards detection system, like oil spill detection [146]. Another real-life example of imbalanced data streams is continuous medical screening[2] for a condition being usually performed on a large population of people without the condition, in order to detect a small minority among them (e.g., HIV prevalence in the USA is ca. 0.4%) or the conversion rates of *online* ads, estimated to be a lie between 10^{-3} to 10^{-6}. Examples can also be found in banking (fraud detection, anti-money laundry, *etc.*) or cybersecurity (e.g., spam filtering, or intrusion detection). It is also worth noting here that financial

[2]T.Fawcett, *Learning from Imbalanced Classes*, 25th August 2017, `https://svds.com/learning-imbalanced-classes/`

or cybersecurity institutions are trying to develop methods of protection against these violations. However, criminals change their attack models to cheat the security measures developed, i.e., the nature of the decision model changes - so we are dealing with the phenomenon called *concept drift*.

Based on the analysis of the literature, it can be seen that the imbalanced data stream classification problem is poorly represented, what is more, most works do not address the issue of the possibility of the concept drift appearing during the operation of the classification model. There are also only a few works that distinguish the differences between the dynamically imbalanced data stream classification problem and a scenario where the prior knowledge about the entire data set is given [160]. This is a result of the additional problems resulting from the lack of knowledge about the class distribution, which are notably present in the initial stages of the data stream classification [242].

The proposed solutions should, therefore, have high adaptability to changing parameters of the classification task, which guarantees, among others, the approach based on classifier ensemble [135]. On the other hand, such methods should take into account the local characteristics of data distributions and the disproportions among the classes. Therefore, the natural candidate seems to be an approach based on the *Dynamic Classifier Selection* (DES). Due to the fact that the dynamic classifier selection is based only on the local neighborhood of query samples, techniques of this type should not be biased in relation to the majority class. Despite this, only a few works attempt to employ these methods to the problem of imbalanced data classification [198, 259].

1.2 Research hypothesis, its aims and goals

This thesis aims to propose effective (regarding the quality of classification as well as computational efficiency) algorithms for the task of classifying highly imbalanced data stream with concept drift occurrence. Additionally, it intends to meet the need to develop new *Classifier Selection* algorithms dedicated to the classification of data with the skewed class distribution. The research hypothesis is as follows:

> *There exist such methods employing data preprocessing and classifier selection that can outperform state-of-the-art classifiers for difficult data classification tasks.*

Aims and goals
In order to confirm the expressed hypothesis, the following goals have been formulated:

1. Developing an ensemble selection algorithm for imbalanced data classification, as well as designing a dedicated combination rule.
2. Proposing a novel distance-based *Dynamic Ensemble Selection* method for imbalanced data classification.
3. Developing a chunk-based ensemble algorithm, aimed specifically for the task of highly imbalanced data stream classification.
4. Designing a novel framework combining *Dynamic Ensemble Selection* and preprocessing techniques for imbalanced data stream classification.
5. Proposing a strategy for learning from drifting data stream under limited access to labels scenario.
6. Evaluating the behavior of the previously proposed data stream classification framework, taking into account the limitation in the label access.
7. Conducting an experimental evaluation of the proposed methods in comparison to *state-of-the-art* approaches.
8. Developing a *Python* Machine Learning library for difficult data stream analysis.

1.3 Thesis structure

Chapter 2 introduces selected topics of pattern classification, with an emphasis on inductive learning and classification task. Classifier ensemble is discussed, including its components, ensemble diversity, and the notion of classifier selection. The problem of difficult data classification is precisely defined, with an emphasis on imbalanced data classification, data stream classification, and limited access to labels. The Python *stream-learn* library for difficult data stream analysis, which was developed during the work on this thesis, is also presented. Chapter 3 presents the ensemble algorithms proposals using classifier selection for imbalanced data classification. The first algorithm employs diversity-based static classifier selection, the second proposition combines base models using a multistage organization, and the third approach proposes *Dynamic Classifier Selection* based on Euclidean distance. Chapter 4 presents proposed algorithms for the classification of unbalanced data streams. The ensemble algorithm employing a classifier selection approach in order to focus on the minority class detection is presented, followed by a novel framework combining dynamic classifier selection and data preprocessing. Chapter 5 deals with the problem of limited access to labels when classifying data streams. First, an algorithm combining *active learning* and random labeling is introduced. Then, the imbalanced data stream classification framework introduced in Chapter 5 is extended with an *active learning* module and evaluated under limited access to labels scenario. Chapter 6 concludes the thesis and presents potential future research directions.

Chapter 2

Selected topics of pattern recognition

This chapter aims to introduce the areas which form the basis of the following thesis and are necessary to properly explain the proposed ideas. First, the basics of pattern recognition will be presented, including the formulation of the classification task, an introduction to classifier ensemble, as well as the notion of diversity and the *Classifier Selection*, with an emphasis on the *Dynamic Ensemble Section*. Then, the subject of difficult data classification will be briefly introduced, including data with a skewed class distribution, data stream classification, as well as scenarios with limited access to labels. Finally, the approach to classifier evaluation for imbalanced and streaming data will be discussed and the developed Python package for difficult data stream analysis will be presented.

2.1 Inductive learning

With the advent of personal computers and the spread of wireless communication, large companies lost their monopoly on generating and storing data. Instead, data is now generated by virtually all internet users in their typical day-to-day activities.

The appearance of a large amount of data introduced problems that cannot be solved with a fixed algorithm containing a sequence of instructions. In such cases, we know the input and we know what the output should be, but we do not know the process that leads to the transition from one to the other. It is difficult especially due to the fact that the process may be influenced by factors changing over time. However, we can try to compensate for this lack of knowledge with the amount of data we have and learn to distinguish between the examples described by the different outputs after their in-depth analysis. To sum up, we want the computer to automatically find an algorithm appropriate for the task at hand [200]. Even if we are not able to completely identify the transformation process in this way, we can construct a useful approximation that, while it does not explain everything, may be sufficient to properly identify some of the patterns present in the data.

This field is known as *machine learning* and uses statistical theory to build models capable of drawing conclusions from known examples [5]. The following thesis focuses, among the different learning methods, on *inductive learning* [174]. In this approach, the learner uses the available examples to generalize hypotheses for the problems in question. Hence, inductive learning algorithms can at best ensure that the output hypothesis fits the concept with respect to the training data. The fundamental assumption of *inductive learning*, formulated by Mitchell [174], states that *"Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples."* In the *inductive learning*, two main types of tasks can be distinguished:

- *Supervised learning*, assuming *a prior* knowledge, which identifies samples from the training set as members of predefined classes in form of the labels [116]. These labels, typically provided by an expert, allow for learning dependencies between the class and data characteristics. Then, learned rules are generalized for the previously unseen data. In the *supervised learning*, one can distinguish a *classification task*, in which the target label is a discrete value, and a *regression* task [5], in which the class is represented by a continuous value.
- *Unsupervised learning*, which assumes that labels cannot be accessed. Therefore, the obtained data is analyzed in order to understand its structure and relations between the problem instances. *Unsupervised learning* consists of the (*i*) task of *density estimation*, where e.g. with the use of *clustering* [199] the unlabeled objects are grouped based on their similarity, and (*ii*) the task of *dimensionality reduction* [232], the methods of which are used to extract and select features for the purposes of classification and visualization [216].

Additionally, we can distinguish *semi-supervised learning* [175], in which at the learning stage the model receives both labeled and unlabeled data. This scenario is typical in cases where labels are not readily available or have a high cost to obtain. In the special case of *semi-supervised learning*, called *active learning* [207], the aim is to determine which of the unlabeled instances, after asking an expert about their labels and adding them to the training set, will be able to improve the system performance to the point of being comparable to the standard *supervised learning* scenario. Another branch of *semi-supervised learning* is *self-labeling* or *self-learning* [230]. In such approaches, a classifier is trained using an initially small number of labeled samples, in order to classify the unlabeled instances. The most confident predictions are added to the training set, which is then used to retrain the model.

This dissertation deals mainly with the notion of *concept learning*, which is a type of *supervised learning*. It involves using training examples to acquire general concepts, which describe some subset of objects. Each concept can be defined as a binary function that divides samples into ones belonging and not belonging to the concept [254]. Mitchell defined *concept learning* as *"Inferring a boolean-valued function from training examples of its input and output"* [174].

2.2 Pattern classification task

As mentioned earlier, the following thesis will focus on supervised learning, and more precisely, on the classification task. The purpose of the classification is to assign a given object to one of the classes predefined in the form of labels, and the process is carried out based on the values of attributes characterizing this object. To formalize this task, we have a feature space denoted by $\mathcal{X}$, where $x \in \mathcal{X}$ is the feature vector representing an object. Assuming, that the feature vector is d-dimensional

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

where $x^{(l)} \in \mathcal{X}^{(l)}$.

Denoting the labels set containing predefined categories as $\mathcal{M} = \{1, 2 \dots, M\}$, a classification algorithm in form of a function Ψ with domain $\mathcal{X}$ and codomain $\mathcal{M}$ assigns a given object to it's category during *classification* process

$$\Psi : \mathcal{X} \to \mathcal{M}. \tag{2.2}$$

This decision is made by the *classifier* with the use of *support functions* which inform about the chance of the object belonging to each class

$$\mathcal{F} = \{F_1, F_2, \ldots, F_M\}. \tag{2.3}$$

Usually, the class of a given x is determined by the highest value of obtained *support function* which is equivalent to the maximum rule

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

Probabilistic Approach

Due to the fact that the classifier's decision is made by applying the maximum rule on the *support function*, the problem of uncertainty of the object's belonging to the class arises. Although any continuous classifier output can be used [67], the main discriminant of the *support function* – associated with probabilistic models – is *posterior* probability [22]. The statistical decision theory is an effective approach to the uncertainty management, which assumes that both the feature vector $x \in \mathcal{X}$ and its class label $j \in \mathcal{M}$ are defined as observed values of random variables pair $(\mathbf{X}, \mathbf{J})$ [75, 254]. The probability distribution of these random variables is given by *prior* class probabilities

$$p_j = P(\mathbf{J} = j), \ j \in \mathcal{M} \tag{2.5}$$

and class-conditional probability density function of $\mathbf{X}$

$$f_j(x) = f(x|j), \ x \in \mathcal{X}, \ j \in \mathcal{M}. \tag{2.6}$$

The main goal when designing a classification system should be to minimize the average misclassification cost, which can be defined on the basis of so-called loss function used to measure the decision cost between the classes

$$L : \mathcal{M} \times \mathcal{M} \to \mathcal{X}, \tag{2.7}$$

where $L(i, j)$ returns the loss associated with the wrong assignment of the object from class j to class i. This allows for formulating the criterion of classification task for the optimal Bayes classifier

$$\min_{\Psi} Risk(\Psi) = Risk(\Psi^*), \tag{2.8}$$

where

$$Risk(\Psi) = E[L(i, j)] = \int\limits_{\mathcal{X}} \sum_{j=1}^{M} L(\Psi(x), j) p_j f_j(x) dx. \tag{2.9}$$

The goal here is to minimize the $Risk(\Psi^*)$, which is defined as the average risk of the classifier Ψ. This allows the so-called conditional risk to be minimized

$$r_i(x) = \underset{\mathbf{J}|x}{E}[L(i,j)] = \sum_{j=1}^{M} L(i,j)p_i(x). \tag{2.10}$$

Which in turn leads to the following decision rule for the optimal Bayes classifier

$$\Psi^*(x) = i \text{ if } \sum_{j=1}^{M} L(i,j)p_j(x) = \min_{k\in\mathcal{M}} \sum_{j=1}^{M} L(k,j)p_j(x), \tag{2.11}$$

where the posterior probability $p_j(x)$ can be calculated from the Bayes formula

$$p_j(x) = \frac{p_j f_j(x)}{\sum\limits_{k=1}^{M} p_k f_k(x)}. \tag{2.12}$$

Considering the popular $0-1$ loss function, which is often used in the practical tasks due to the inability to assess the loss values

$$L(i,j) = \begin{cases} 0 \text{ if } i=j \\ 1 \text{ if } i \neq j \end{cases}, \tag{2.13}$$

the following decision rule aiming to minimize the misclassification probability of the optimal Bayes classifier Ψ^* can be obtained

$$\Psi^*(x) = i \text{ if } p_i(x) = \max_{k\in\mathcal{M}} p_k(x). \tag{2.14}$$

As the defined loss function is related to the class with the highest *posterior* probability and the conditional risk is defined as the probability of misclassifying a sample x, the risk of misclassification probability can be averaged

$$Risk(\Psi^*) = P_{err}(\Psi^*) = \sum_{j=1}^{M} p_j \int\limits_{D_j} f_j(x)dx = 1 - \int\limits_{X} \max_{j\in\mathcal{M}} p_j f_j(x)dx = 1 - P_{acc}(\Psi^*). \tag{2.15}$$

Overfitting

To build a classification model, the $\mathcal{LS}$ training set is used, which groups the observations from a given domain in the form of pairs

$$\mathcal{LS} = \{(x_1, j_1), (x_2, j_2), \ldots, (x_N, j_N)\}, \tag{2.16}$$

where x_k denoted the feature vector of the k-th learning pattern, j_k is its correct label and N is the cardinality of $\mathcal{LS}$. Each element in this set corresponds to a single instance of a problem and its proper class.

Two types of errors can be observed in the classification task

- The training error, which is defined as the proportion of incorrectly classified objects from the training set to its cardinality

$$P_{err}^{\mathcal{LS}}(\Psi) = \frac{\sum\limits_{k=1}^{N} [\Psi(x_k) \neq j_k]}{|\mathcal{LS}|}. \tag{2.17}$$

- The real error (also known as the generalization error), which is defined as the number of misclassified objects drawn from the general population

$$P_{err}(\Psi) = \int\limits_{X} P(\Psi(x) \neq i|x) f(x) dx. \tag{2.18}$$

Overfitting is a phenomenon related to the loss of the classifier's ability to generalize the acquired knowledge. This means that the model, instead of extracting knowledge from a given data set, begins to remember individual instances. In this case, due to too much training complexity or an insufficient number of examples, the learner is not able to correctly predict the labels of instances that were not present in the training process. Due to this phenomenon, the classification accuracy on previously unseen data decreases, while the accuracy on training data increases consistently.

In practice, we can say that the classifier Ψ overfits the learning data $\mathcal{LS}$ if there exists another classifier Ψ' such that

$$P_{err}^{\mathcal{VS}}(\Psi) > P_{err}^{\mathcal{VS}}(\Psi') \; and \; P_{err}^{\mathcal{LS}}(\Psi) < P_{err}^{\mathcal{LS}}(\Psi'), \tag{2.19}$$

where $P_{err}^{\mathcal{VS}}$ is an error on the validation dataset ($\mathcal{VS}$) which contains a set of examples not presented during the training procedure [174]. The classifier error can be broken down into three components [125]:

- Error, lower bounded by the error of the optimal Bayes classifier, that is specific to the problem and cannot be eliminated.
- The error related to bias resulting from the assumptions made by the model based on the training data.

- The error related to variance related to training data.

Bias is defined by Mitchell as *"(...) the set of assumptions that the learner uses to predict outputs given inputs that it has not encountered.*[1] ", and mathematically it can be described as the difference between the actual and expected outputs. Variance, on the other hand, determines how much the model's predictions vary depending on the training data used. In the case of high bias, the assumptions are too simple and the model misses the relevant relationships present in the data, which results in *underfitting*. High variance causes the model to fit too closely to the training set, which causes the previously described *overfitting* phenomenon. *No free lunch theorem* [251], formulated by Wolpert, tells us that there is no such thing as one universal machine learning algorithm that can do best for all the problems encountered, as each has its own domain of competence. These competencies result from the learner's bias, which, according to the *Ugly Ducking theorem* [75], is necessary to generalize knowledge and carry out the classification process. Therefore, we are dealing with a *bias-variance dilemma* [93], in which, on the one hand, assumptions are necessary to train the classifier - which increases the bias, and on the other hand, reducing the bias increases the demand for samples and thus increases the variance. Propositions for dealing with this problem include approaches such as comparative study of models using cross-validation, penalizing model complexity based on augmented error function [5], selecting models based on their complexity [235] and trying to find the best model based on so-called Minimum Description Length (MDL) [193].

Description of selected classifiers

Let's introduce the classification algorithms chosen from the five different families, which will be used for experiments performed later in this thesis.

- *Bayesian Classifiers*, family of probabilistic classifiers based on Bayes' theorem [15]. Common examples here are the *Naïve Bayes classifier*, which simplifies the conditional probability by assuming strong independence between the problem's features.

- *Minimal Distance Classifiers*, where the most popular example is the *k-Nearest Neighbors* (kNN) algorithm [56]. Here, the sample classification is performed by a majority vote of its k nearest neighbors found in the learning set. The neighborhood is determined based on the chosen distance metric, which usually is the Euclidean distance. kNN is an example of a *lazy* learner, which delays the generalization process until the prediction phase [170].

[1]Tom M. Mitchell, Machine learning, McGraw-Hill, New York, 1997.

- *Rule-Based Classifiers*, using the *indirect learning* approach, in which the decision tree is first trained and then converted to rules easily interpretable for humans [254]:
  - *Classification and Regression Decision Tree* (CART) [32], which constructs binary decision tree and employs the Gini index as the impurity measure for assigning features to the nodes.
  - *Hoeffding Tree* (HT) or *Very Fast Decision Tree* (VFDT) [73], which processes each sample in constant time and memory. It uses Hoeffding bounds [110] to ensure, that the output obtained by the incremental learner is asymptotically nearly identical to that of conventional model.
- *Neural Networks*, defined as structures composed of a number of artificial neurons, which interact with each other on the basis of weights. McCulloch and Pitts formulated the first model of simple artificial neuron capable of performing basic logical operations [171], while Rosenblatt proposed the *perceptron*, which was able to perform classification based on the sum of the weighted inputs and activation function [195].
- *Support Vector Machines* (SVM) [46], based on the concept introduced and then expanded by Vapnik [233–235]. They consist of a set of binary supervised learning methods, with a goal to form the hyperplane separating data points into two sets by mapping them into a high-dimensional space.

2.3 Classifier ensemble

The following section presents the concept of a classifier ensemble. The stages of base model generation and combination are discussed, while special emphasis is placed on the optional stage of classifier selection - especially the *Dynamic Ensemble Selection*.

Components of Multiple Classifier Systems

One of the most popular and still actively developed approach to classification is one in which, instead of using a single learner, we employ multiple classification models, and then we combine their decisions in order to obtain the final output. The aim here is to take advantage of the strengths of each combined classifier and their domain of competence. Deserathy and Sheela first applied this approach in 1979 [66] when they combined k-NN and a linear classifier, and since then many studies have demonstrated the effectiveness of using multiple models instead of a single one [206]. Such an approach is known as a *classifier ensemble* or a *multiple classifier system* (MCS) [150] and its main components organized in the parallel topology [254], which is by far the most common, are depicted in Figure 2.1.

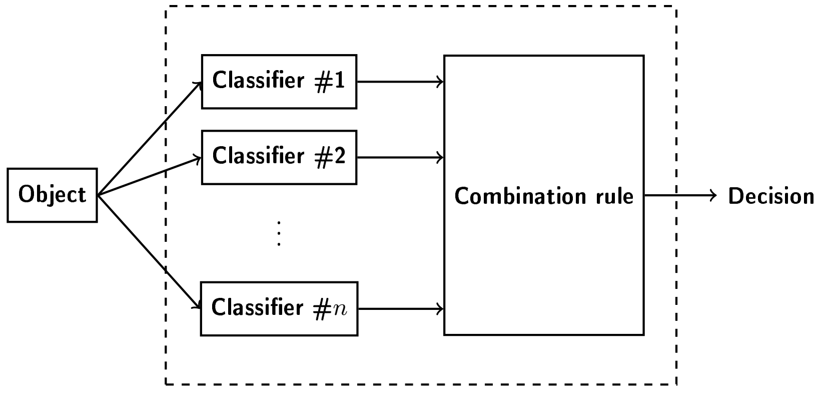


Figure 2.1: *Parallel topology of a classifier ensemble.*

A multiple classifier system consists of three steps [33]: i) Generation, ii) Selection and iii) Combination (also known as Fusion or Aggregation). It should be noted that the selection can be performed as a separate process or in conjunction with the combination block. It is also entirely optional and not used by some of the ensemble algorithms.

The purpose of the generation stage is to train a *pool of classifiers* $\Pi = \{\Psi_1, \Psi_2, \ldots, \Psi_n\}$, where n is a number of base models. The two most important determinants of a good classifier ensemble are that the base models are both diverse (as there is no reason to combine classifiers offering the same output [68]) and accurate, which in this case means that they perform better than the random classifier.

2.3.1 Ensemble diversity

As mentioned above, one of the determinants of a valuable classifier ensemble is the high diversity of its base models, therefore the question of how to measure this diversity arises. According to Kuncheva [150], there are two styles of measuring the classifier pool diversity:

- Pairwise diversity measures calculates diversity between each pair of classifiers and then average the results to obtain value for the entire ensemble. For a classifier pool consisting of n models there are $\frac{n(n-1)}{2}$ values of pairwise diversity. Examples of such measures include Q-statistic [266], disagreement measure [108, 213] and double-fault measure [95].

- Non-pairwise measures take into account all the learners in the pool and offer a single diversity value for the entire ensemble. Among these types of measures, we can distinguish the entropy measure E [61] and Kohavi-Wolpert variance [125].

Diversity is one of the key factors for generating a valuable classifier ensemble, but the main problem is how to measure it. Let us present the selected diversity measures:

The entropy measure E [61] is defined as

$$E(\Pi) = \frac{1}{N}\sum_{j=1}^{N}(\frac{1}{n-[n/2]})min\{l(x_j), n-l(x_j)\}, \tag{2.20}$$

where N is the number of instances, n stands for the number of base models in the ensemble and $l(x_j)$ denotes the number of classifiers that correctly recognize x_j. E varies between 0 and 1, where 0 indicates no difference and 1 indicates the highest possible diversity.

Kohavi-Wolpert variance [125] is defined as

$$KW(\Pi) = \frac{1}{Nn^2}\sum_{j=1}^{N} l(x_j)(n-l(x_j)). \tag{2.21}$$

The higher the value of KW, the more diverse the classifiers in the ensemble. Also, KW differs from the averaged disagreement measure Dis_{av} by a coefficient, i.e.,

$$KW(\Pi) = \frac{n-1}{2n}Dis_{av}(\Pi), \tag{2.22}$$

Measurement of interrater agreement k [80] [69]

$$k(\Pi) = 1 - \frac{\frac{1}{n}\sum_{j=1}^{N} l(x_j)(n-l(x_j))}{N(n-1)\bar{p}(1-\bar{p})}, \tag{2.23}$$

where $\bar{p}$ is average individual classification accuracy

$$\bar{p} = \frac{1}{Nn}\sum_{j=1}^{N}\sum_{k=1}^{n} i_{j,k}, \tag{2.24}$$

where $i_{j,k}$ is an element of an N-dimensional binary vector $i_k = [i_{1,k}, \ldots, i_{N,k}]^T$ representing the output of a classifier Ψ_k, such that $i_{j,k} = 1$, if Ψ_k recognizes x_j correctly, and 0 otherwise. Measurement of interrater agreement k varies between 1 and 0, where 1 indicates complete agreement and 0 indicates the highest possible diversity.

Table 2.1: *A table of the relationship between a pair of classifiers.*

	Ψ_k correct (1)	Ψ_k wrong (0)
Ψ_i correct (1)	N^{11}	N^{10}
Ψ_i wrong (0)	N^{01}	N^{00}

The averaged Q statistics [266] over all pairs of classifiers is given as

$$Q_{av}(\Pi) = \frac{2}{n(n-1)} \sum_{h=1}^{n-1} \sum_{k=h+1}^{n} Q(\Psi_h, \Psi_k), \tag{2.25}$$

where

$$Q(\Psi_h, \Psi_k) = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}, \tag{2.26}$$

and N^{ab} is the number of elements x_j for which $i_{j,h} = a$ and $i_{j,k} = b$. Relationship between a pair of classifiers is denoted according to Table 2.1. Q varies between -1 and 1. Classifiers that recognize the same objects correctly will have positive values of Q, and those which commit errors on different objects will render Q negative.

The averaged disagreement measure [108] over all pairs of classifiers is given as

$$Dis_{av}(\Pi) = \frac{2}{n(n-1)} \sum_{h=1}^{n-1} \sum_{k=h+1}^{n} Dis(\Psi_h, \Psi_k), \tag{2.27}$$

where

$$Dis(\Psi_h, \Psi_k) = \frac{N^{01} + N^{10}}{N^{11} + N^{10} + N^{01} + N^{00}}. \tag{2.28}$$

The averaged disagreement measure is the ratio between the number of observations on which one classifier is correct and the other is incorrect to the total number of observations. Dis varies between 0 and 1, where 0 indicates no difference and 1 indicates the highest possible diversity.

It should be noted, however, that despite the multitude of available measures, none of them can be considered best suited to minimize the classification error. The only recommendation can be made on the basis of the ease of interpretation of a given measure [151].

Ensuring classifier diversity

Another problem that arises is how to ensure the diversity of the generated pool of classifiers. According to the literature, this issue can be approached in three different ways [96, 150]:

- Classifiers can be trained on different input data. This can be done by using different data partitions, e.g. through bootstrapping approaches such as *Bagging* [30], which creates new training sets, based on the original one, for each base model through sampling with replacement. Another approach of this type is *Boosting* [81, 83] which in the case of its most well known *AdaBoost* algorithm [82] generates subsequent training sets by increasing the probability of drawing instances that have been incorrectly classified. Walmsley et al. proposed a classifier pool generation method based on *Bagging*, in which the probability of instance selection during the resampling corresponds to the instance hardness [237]. Online pool generation method for generating locally accurate classifier pool in difficult regions of feature space was proposed by Souza et al. [217]. Jamalinia et al. proposed the *Ensemble-based Artificially Generated Training Samples* (EBAGTS) algorithm, which manipulates training samples based on error-prone instances and feature space regions [117]. Hido et al. proposed the *Roughly Balanced Bagging* (RBB) [106], extensively studied by Lango and Stefanowski [154], which uses sampling to balance the class distribution across all bootstraps for the imbalanced data classification task. The linear Modification of the *AdaBoost* algorithm was proposed by Burduk [44]. Burduk and Bozejko modified the *Gentle AdaBoost* algorithm [84] on the basis of scaled distance from the decision boundary [45].

  The base models can also be trained using different subsets of the problem features. This approach to diversification is known as *Random Subspace* [107, 108] (also called *Attribute Bagging* [35] or *Feature Bagging* [157]) and is used, among others, by the *Random Forrest* algorithm to generate trees fitted on randomly chosen attributes [31]. Algorithms based on *Random subspace* are still quite popular and constantly find their way into new applications. Wang et al. proposed the *Deep Random Subspace Ensemble* (DRSE), which integrated *Random Subspace* with *deep learning* methods [240]. The *Random Subspace based Ensemble Sparse Representation* (RS_ESR) algorithm, which introduced the feature resampling into sparse representation model, was proposed by Gu et al. [97]. Blaszczykowski and Stefanowski proposed the *Ordinal Consistency Driven Feature Subspace Aggregating* (*coFeating*), which construct local classifiers in chosen regions of the feature space [24]. Blaser and Fryzlewicz improved the ensemble diversity by generating each base classifier using a randomly rotated feature space [23]. There are also approaches that train base classifiers on features derived from many different feature extraction methods, which have been successfully used in the task of face image classification [12].

  Another way is to select classifiers from the generated pool, which assumes that each of the base models is an expert in a certain region of the feature space. The selection can be conducted either in a static or dynamic fashion. In the static approach, the models are selected once during the training phase and the same ensemble/single classifier is used to classify all unknown examples [270]. In the dynamic approach, a separate ensemble or a single model is selected for each unknown problem instance [59]. Such approaches are one of the main topics of this thesis and will be described in detail later.

- Classifiers can also be trained to recognize only some of the problem classes. In this case, the combination method used should be able to recover the entire set of labels before a final decision is made. These types of approaches are based on the fact that any multiclass problem can be broken down into a number of binary problems [228] and propose different methods to build a multiclass classifier by combining two-class models [71]. Some of the well known binarization strategies are:
  - *One-vs-One* (OVO) [103], which trains a binary classifier for each pair of classes.
  - *One-vs-All* (OVA) [192], which train a binary classifier for each class, considering all remaining classes as a majority one.
  - *One-Against-Higher-Order* (OAHO) [177], which sorts the classes in descending order by the number of samples and iterates starting from the largest one. A binary classifier is generated for the current class and all remaining classes with less cardinality.
  - *All-and-One* (A&O) [92], which combines OVO and OVA.
  - *The Error Correcting Output Codes* (ECOC) [70], which encodes each class with a code-word in order to obtain the distance between classes.
- Finally, ensemble diversity can be ensured by creating a pool containing different classification models. This can be done, for example, by training different machine learning algorithms (heterogeneous ensemble) on the same input data [231]. Another method may be to use a single classification algorithm (homogeneous ensemble), but differentiate it by modifying its parameters. An example of such an approach is the modification of the initial weights of neural network [112]. Approaches combining heterogeneous ensembles with data-level diversification for real-life applications, such as credit scoring, are also gaining popularity [258].

2.3.2 Combination rule

During the combination stage, the answers obtained by each of the base classifiers are processed, using the chosen *combination rule* [122], in order to reach a final decision. These rules take advantage of the fact, that the base classifiers outputs have a clear interpretation and may be represented by class labels, distances or confidences (probabilities) [76]. Usually, the combination process is be based on the class labels returned by the models or on their *support functions* (Equation 2.3 on p. 22).

One of the most common approaches to combination based on class labels is *voting*. In its simplest version, called *majority voting*, the instance is assigned to the class that was most often indicated by the base models

$$\Psi(x) = \underset{i \in \mathcal{M}}{argmax} \sum_{k=1}^{n} \left[\Psi_k(x) = i\right], \tag{2.29}$$

where [ ] denotes the Inverson's bracket.

There is also *weighted voting*, which introduces weight w_k for each of the k base classifiers in such a way that they may have different influence on the final decision

$$\Psi(x) = \underset{i \in \mathcal{M}}{argmax} \sum_{k=1}^{n} \left[\Psi_k(x) = i\right] w_k. \tag{2.30}$$

Another popular approach to the labels-based classifier combination is known as *Stacked Generalization* (*(Stacking)*) [250]. Here, the combination rule (also known as meta-classifier or meta-level classifier) is trained based on the predictions made by base models. In order to reduce the possibility of the meta-classifier overfitting, the dataset used for combination rule training should be excluded from the dataset used for generating the base classifiers. Usually, stacking employs a heterogeneous classifier pool in order to assure their diversity.

When the decision is made on the basis of the *support functions*, a common approach is to use the *aggregation* (also called *accumulation* or *the sum rule*) of supports

$$\Psi(x) = i \quad if \quad F_i(x) = \underset{k \in \mathcal{M}}{max} F_k(k, x), \tag{2.31}$$

where

$$F_l(x) = \sum_{l=1}^{n} w_l F_{l,i}(x) \quad and \quad \sum_{i=1}^{n} w_l = 1, \tag{2.32}$$

where $F_l(x)$ denotes the support function for the i^{th} class of the l^{th} classifier, and w_l is a classifier weight.

These weights are usually static [150], but their values may also change depending on the classifier and label [257] or be a function of feature vector [202]. Regardless of whether the combination is based on the labels or support functions, there exists various possibilities of weight assigning [255]:

- Weights dependent on classifier – a traditional approach where each the l-th classifier is weighted by the value w_l.
- Weights dependent on classifier and feature vector – where weight $w_l(x)$ is assigned to the l-th classifier and a given sample x.
- Weights dependent on classifier and class number – where value $w_{l,i}$ is assigned as weight to the l-th classifier and the i-th problem class.
- Weights dependent on classifier, class number, and feature vector – where weight $w_{l,i}(x)$ is assigned to the l-th classifien, a given sample x and the $i-th$ class.

Besides *the sum rule* or its weighted equivalent, base classifiers can be aggregated using simple operators such as [76]:

- *The product rule*, which corresponds to *the sum rule* for small deviations in the classifier outputs. Theoretically it performs well if the base models are independent, which unfortunately is an unrealistic assumption.
- *The maximum rule*, which selects the classifier most confident in its own predictions and can be interpreted as a kind of classifier selection. This rule is very sensitive to overfitting.
- *The minimum rule*, which chooses the classifier with the least objection to the certain class.
- *The median rule*, which is similar to *the sum rule* but may give more robust results.

Another approach worth mentioning is the *Mixture of experts* [114, 115], which divides the problem space into a number of subspaces and trains an expert learner of each of them. This process is managed using a *gating function* which is trained together with the experts and then used to dynamically compute weights for base classifiers taking into account their local competencies.

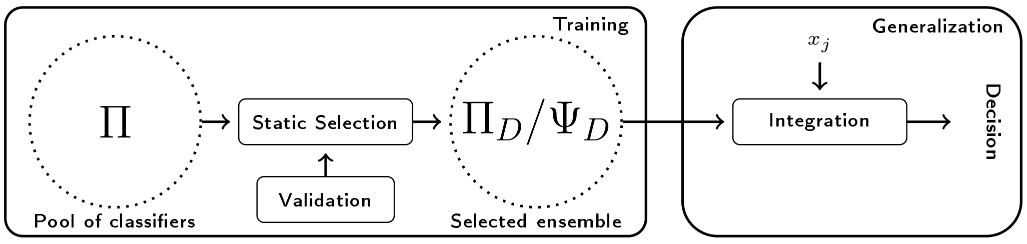
Alternative combination proposal is a multiple-stage organization, which was briefly mentioned by Ho et al. [109] and described in detail by Ruta and Gabrys [201], where authors refer to such systems as a multistage organization with majority voting (MOMV) since the decision at each level is given by majority voting. Initially, all outputs are allocated to different groups by permutation and majority voting is applied for each group producing single binary outputs, forming the next layer. In the next layers, exactly the same way of grouping and combining is applied with the only difference being that the number of outputs in each layer is reduced to the number of groups formed previously. This repetitive process is continued until the final single decision is obtained.

2.3.3 Classifier selection

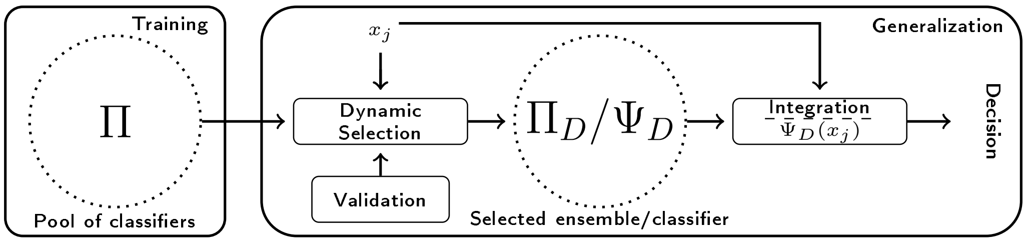
Classifier selection employs the *overproduce-and-select* approach, in which the models used in the classification process are selected on the basis of their local competencies. There are two approaches to the selection process:

- *Static selection*, presented in Figure 2.2 a, in Figure in which the selection process of a classifier or an ensemble is performed during the training phase, based on the selection criterion estimated in the validation dataset. Then, this exact ensemble is used in the generalization phase to predict the labels of all test samples. Classifier diversity and classification accuracy are among the most common selection criteria. Among the well-known algorithms implementing this approach, *Classifier and Selection* proposed by Kuncheva [148] can be distinguished. Another example is the approach proposed by Jackowski et al. [113] called *Adaptive Splitting and Selection*, which uses an evolutionary algorithm to find the best partitioning of the feature space and matches each cluster with the most fitting ensemble.
- *Dynamic selection*, depicted in Figure 2.2 b, where the discriminant ability of classifiers is assessed in the local region of competence for each unknown example separately. Then, based on these competencies, the selection is performed individually for classifying each of these samples. Since the *Dynamic Classifier Selection* is one of the main topics of this thesis, it is described in more detail below.

Dynamic Selection methods can select either a single model (*Dynamic Classifier Selection* - DCS) or an ensemble of classifiers (*Dynamic Ensemble Selection* - DES), with the latter being recognized as a very promising direction in ensemble learning [59]. DES selects the best classifiers for each test instance based on the notion of competence, which is usually estimated in the local region of competence containing, e.g., the k-Nearest neighbors of the given sample. This region is formed using the *dynamic selection dataset* (DSEL) composed of labeled samples from either the training or validation set. This is based on the assumption that each of the base classifiers is an expert in a different region of the feature space.



(a) *Static selection*



(b) *Dynamic selection*

Figure 2.2: *Static and dynamic classifier selection procedure.*

The classification of each unknown sample by DES involves three steps:

- Definition of the region of competence; that is, how to define the local region surrounding the unknown sample, in which the competence level of the base models is estimated. This local region of competence is found in the dynamic selection dataset (DSEL), which is usually part of the training set.
- Defining the selection criterion later used to assess the competence of the base classifiers in the local region of competence (e.g., accuracy or diversity).
- Determination of the selection mechanism deciding whether we choose a single classifier or an ensemble.

Previous work related to the imbalanced data classification using classifier ensembles and DES involves various approaches. Ksieniewicz proposed an *Undersampled Majority Class Ensemble* (UMCE) [140] employing different combination rules and pruning, based on a k-fold division of the majority class to divide a single imbalanced problem into many balanced ones. Chen et al. [51] presented the *Dynamic Ensemble Selection Decision-making* (DESD) algorithm to select the most appropriate classifiers using a weighting mechanism to highlight the base models that are better suited for recognizing the minority class. Roy et al. combined preprocessing with dynamic ensemble selection to classify both binary and multiclass stationary imbalanced datasets [198]. *Randomized Reference Classifier*, which produces supports for each class that are realizations of random variables with the beta distributions, was proposed by Woloszynski and Kurzynski [249]. Lysiak et al. [165] showed that employing diversity measures during the classifier selection leads to smaller ensembles but does not improve the classification accuracy. META-DES.Oracle, which uses feature selection and meta-learning over numerous datasets to improve the selection process, was proposed by Cruz et al. [58]. Oliveira et al. [183] described a two-step ensemble forming using a pre-selection mechanism. Zyblewski et al. [277] proposed the *Minority Driven Ensemble* algorithm, which employs a dynamic classifier selection approach to exploit local data characteristics for imbalanced data streams classification. The proposal of combining preprocessing and *Dynamic Ensemble Selection*, which is the basis of research carried out in this work, was presented by Zyblewski et al. [280]. Pinagé et al. proposed a concept drift detection method based on dynamic classifier selection [188].

We may also consider the following DES strategies based on oracle information, which will be used later in conducted experiments:

- KNORA-Eliminate (KNORA-E) [123], which creates an ensemble consisting only of the local oracles, i.e., models that classify correctly all data samples located in the local region of competence. In the case where no classifier is selected, the size of competence region is reduced by removing the farthest neighbor and the search for oracles is repeated,
- KNORA-Union (KNORA-U) [123] makes the decision based on weighted voting, where each selected classifier has a number of votes proportional to the number of correctly predicted samples in the local region of competence.
- DES-KNN [214] ranks individual classifiers according to their prediction performance and then the fixed number of the best classifiers are first selected. The final ensemble is formed based on the fixed number of the most diverse preselected individuals.
- DES-Clustering [214] employs the *k-Means* to define DESL, then the most accurate and diverse classifiers ale selected for the ensemble.

Additionally, as the reference methods, two *Dynamic Classifier Selection* algorithms will be used:

- *Modified Classifier Ranking* (*Rank*) [203, 252] uses for classification such an individual classifier which classifies correctly the highest number of consecutive samples in the region of competence.
- *Local classifier accuracy* (LCA) [252] selects for classification such an individual classifier which correctly classifies the higher number of samples within the local

region, but considering only those examples where the classifier predicted the same class as the one it gave for the test instance.

Ensemble pruning

Another concept, closely related to the classifier selection, is known as ensemble pruning. Let us first present the ensemble pruning taxonomy proposed in [270]:

- *Ranking-based pruning* chooses a fixed number of the best ranked individual classifiers according to a given metric (as kappa statistics) [169].
- *Optimization-based pruning* solves the problem of choosing individual classifiers as an optimization task. Because the number of base models is typically high, therefore heuristic methods [202], evolutionary algorithms [272] or cross-validation based techniques [65] are usually used.
- *Clustering-based pruning* looks for groups of base classifiers, where individuals in the same group behave similarly while different groups have large diversity. Then, from each cluster, the representative is selected, which is placed in the final ensemble.

As the following thesis partially deals with employing clustering-based classifier ensemble pruning methods to improve the predictive performance of combined classifiers then let us briefly present the main works related to this field. Clustering-based pruning consists of two steps. In the first one, base models are grouped into several clusters based on a criterion, which takes into consideration their impact on the ensemble performance. For this purpose, various clustering methods were used, such as hierarchical agglomerative clustering [96], deterministic annealing [10], k-Means clustering [85] [156] and spectral clustering [267]. Most of those methods employ some kind of diversity-based criteria. Giacinto et al. [96] estimated the probability that classifiers do not make coincident errors in a separate validation set, while Lazarevic and Obradovic [156] used the Euclidean distance in the training set. Kuncheva proposed employing a pairwise diversity matrix for hierarchical and spectral clustering methods [150].

In the second step, a prototype base learner is selected from each cluster. In [10] a new model was trained for each cluster, based on clusters centroids. In [96] Giacinto et al. chosen the classifier, which was the most distant to the rest of clusters. In [156] models were iteratively removed from the least to the most accurate. The model with the highest classification accuracy was chosen in [85].

The last issue is the choice of the number of clusters. This could be determined based on the performance of the method on a validation set [85]. In the case of fuzzy clustering methods, indexes based on membership values and data set or statistical indexes can be used to automatically select the number of clusters [132].

2.4 Difficult data classification

The following section aims to discuss the notion of difficult data classification, focusing on the skewed class distribution, data stream, and the case of limited label access, strongly associated with *active learning.* Dealing with these problems, especially in cases of their simultaneous occurrence, is the main focus of this dissertation. As mentioned in the introduction, these three scenarios are not the only definitions of difficult data. However, issues such as data heterogeneity, high dimensionality, a high number of classes, data incompleteness, or low or very high cardinality of the learning set are not dealt with in this thesis, therefore they are not covered in a longer description.

2.4.1 Imbalanced data

Most of the classification algorithms assume that there are no significant disproportions among instances from different classes. Nevertheless, in many practical tasks, we may observe that examples from one class (so-called *majority class*) significantly outnumber the objects from remaining classes (*minority class*). This disproportion, in the case of binary problems, is often represented by the *Imbalance Ratio*, which describes how many majority class samples are there per one minority class sample. Most of the traditional classifiers have a bias in favor of the majority class. However, more often, the minority class is more interesting because misidentification of an instance belonging to it is usually much more expensive than assigning an instance from the majority class to the minority one. A good example is an undetected fraud that would be more expensive than the cost of additional analysis of a correct transaction classified as a fraudless transaction. Such a problem is known as imbalanced data classification [224, 245], where an unequal number of instances from the examined classes plays a key role during the classifier learning. Various approaches have been proposed in the literature to tackle this challenging difficulty embedded in the nature of data. Usually, the researchers are focusing on maximizing the correct minority class classification. At the same time, the performance of the majority class cannot be neglected.

In the case of imbalanced data classification, the disproportion between the different classes is not the sole issue of learning difficulties. One may easily come up with an example where the instance distributions from different classes are well-separated. Proposing an efficient classifier for such a task is not a challenge. Unfortunately, instances from the minority class often form clusters of an unknown structure that are scattered [178]. An additional complication comes from the fact that during learning, the number of instances from the minority class may not be sufficient enough for the learning algorithm to acquire the appropriate generalization level, which in effect can cause *overfitting* [54]. All those problems remain the focus of intense research [43, 49, 147].

Methods for imbalanced data classification can be divided into three main groups, i.e. data preprocessing methods, inbuilt mechanisms and hybrid methods [163].

Data preprocessing methods. This approach focuses on reducing the number of objects in the majority class (*undersampling*) or generating new objects of the minority class (*oversampling*). These mechanisms have the objective of balancing the number of instances from considered classes. For oversampling, new instances are random copies of existing ones (Random Oversampling [13]), or they are generated in a guided manner. The most popular method is *Synthetic Minority Oversampling Technique* (SMOTE) [49] algorithm, which creates new instances based on existing ones by slightly modifying the values of their attributes. As a result, new artificial examples that are compatible with the minority class distribution are generated. Other oversampling methods are ADASYN [104], that also takes into consideration the object difficulties, or RAMOBoost [52]. Unfortunately, methods like SMOTE may lead to changes in the characteristic of the minority class. Consequently, it may result in the classifier *overfitting*. Several modifications of SMOTE have been proposed that are able to identify the instances to be copied in a more intelligent fashion such as *Borderline*SMOTE [100]. It generates new instances from the minority class close to the decision border. *Safe-Level* SMOTE [43] and LN-SMOTE [167] reduce the probability of generating synthetic instances of the minority class in areas where the predominant objects are that of the majority class. SMV-SMOTE employs SVM classifier in order to generate new examples considering it;s support vectors [182]. Among other propositions are: RBO [130] and CCR that enforce instances from the majority-class to be relocated from the areas where the minority-class instances are present [131].

Methods of *undersampling* are built around the idea of randomly removing the instances from the majority-class or removing them in such a way that the quality of the classifier is not disrupted. The most basic method, *Random Undersampling* (RUS) [13], achieves the class balance by random elimination of the majority class intances. *Condensed Nearest Neighbors* (CNN) [102] removes the majority class samples that are close to the decision boundary using 1-NN rule. *Edited Nearest Neighbors* (ENN) [248] computes three nearest neighbors of each instance and a given sample is removed if it belongs to the majority class and is missclassified by its three neighbors. *Neighborhood Cleaning Rule* (NCL) [155] removes samples, for which labels obtained based on ENN rule for three and five neighbors are different. *Tomek's modification of Condensed Nearest Neighbor* (TL) [229] performs guided undersampling using two *Tomek Links*, dedicated for majority and minority class. *One Sided Selection* (OSS) [147] detects *Tomek Link* using 1-NN and then removes all majority samples embedded in it. *Undersampling Based on Clustering* (SBC) [263] divides data into clusters and then, based on the *Imbalance Ratio*, removes samples from the majority class clusters.

Inbuilt mechanisms. In this approach, existing classification algorithms are adapted for imbalanced problems ensuring balanced accuracy for instances from both classes. Two of the most popular areas of research of these methods are using one-class classification [118], usually known as learning without counterexamples. They aim to learn the minority class decision areas, and because of the frequently assumed regular, closed shape of the decision borders is adequate to the clusters created by minority classes [137]. The disproportion between the number of instances in classes is then omitted. Another approach is the (*cost-sensitive*) classification, where the algorithm takes into account the asymmetrical loss function that assigns a higher cost to misclassification of an instance from a minority class [105, 163, 271]. Unfortunately, such methods can cause a reverse bias towards the minority class. There also exists a *cost-sensitive* approach to classifier selection. However, the algorithms proposed so far are based almost solely on static ensembles such as *cost-sensitive trees ensemble* [138], ensemble methods based on ROC space [16, 74], or *cost-sensitive Boosting* [225]. There is a clear lack of DYNAMIC ENSEMBLE SELECTION methods taking into account the different costs of problem classes. Therefore, the proposal of such methods might present another interesting challenge. Worth noting are methods based on ensemble classification [253], like SMOTE*Boost* [50] and *AdaBoost.*NC [241] or *Multi-objective Genetic Programming Ensemble* [17].

Hybrid methods. They combine the advantages of methods using data preprocessing with the classification methods as well ass different approaches to data preprocessing. The most popular category is the hybridization of *undersampling* and *oversampling* with ensemble classifiers [86]. This approach allows the data to be independently processed for each of the base models. Batista et al. proposed two hybrid methods based on the SMOTE oversampling algorithm [13]. SMOTE-ENN combines SMOTE with *Condensed Nearest Neighbor*, which is used to filter noisy items and remove samples from both classes before applying the oversampling algorithm. SMOTE-TL uses SMOTE to generate synthetic minority class instances and then detects and removes samples composing *Tomek Links*. Stefanowski and Wilk proposed the *Selective Preprocessing of Imbalanced Data* (SPIDER) [220], which combines local minority class oversampling with filtering of difficult samples from the majority class. Napierala et al. then extended this idea and introduced the SPIDER2 algorithm [180], which detects noisy samples from both minority and majority class. Majority noisy samples are then relabeled or removed, while the minority noise samples are replicated. *Adaptive Oversampling Technique Based on Data Density* (ASMOBD), which combines oversampling with SELF-LABELING based on the instance difficulty, was proposed by Wang et al. [244]. Yang et al. introduced a hybrid optimal ensemble classifier framework combining density-based undersampling with multi-objective optimization algorithm [261]. Zhaot et al. presented the *Weighted Hybrid Boosting* (*WHMBoost*) algorithm consisting of two base classifiers and two weighted data preprocessing methods

Metrics

The evaluation criterion plays an extremely important role in the process of evaluating the performance of the pattern recognition algorithm. This thesis focuses on the binary classification task, for which all metrics are based on the confusion matrix shown in Table 2.2.

Table 2.2: *The confusion matrix for binary classification.*

	Positive (1)	Negative (0)
Positive (1)	TP	FP
Negative (0)	FN	TN

Traditionally, the accuracy score is used to assess the performance of classification algorithms. Unfortunately, in the case of imbalanced data classification, it is not adequate and informative, as it does not distinguish correctly classified objects of the majority (negative) and minority (positive) class. Therefore, if the minority class we are interested in constitutes, for example, 3% of all instances in a given problem, assigning all of them to the majority class will result in an accuracy score of 97% [166].

$$accuracy(\Psi, \mathcal{VS}) = \frac{TP + TN}{TP + FN + FP + TN} \tag{2.33}$$

Therefore, the evaluation in the case of imbalanced data must be carried out using dedicated metrics that take into account the class distribution. Among these metrics, we can distinguish three base metrics, as well as multiple aggregated metrics:

Recall (also known as *sensitivity* or TPR) [190], which represents the classifier's ability to recognize minority (positive) class objects. It tells us what percentage of minority class instances were detected.

$$recall(\Psi, \mathcal{VS}) = \frac{TP}{TP + FN} \tag{2.34}$$

Miss rate (also known as FNR) [190], which is the inverse of *recall* and tells what percentage of objects belonging to the minority class has not been recognized.

$$miss\,rate(\Psi, \mathcal{VS}) = \frac{FN}{FN + TP} \tag{2.35}$$

Specificity (also knows as TNR) [190], which is equivalent to recall for the majority (negative) class. In the case of problems with the dynamically changing prior class probabilities (including swapping the majority and minority class), an exchange of the *recall* and *specificity* values can be observed.

$$specificity(\Psi, \mathcal{VS}) = \frac{TN}{TN + FP} \tag{2.36}$$

Fallout (also known as FPR) [190], which is the inverse of *specificity* and informs about the percentage of majority class objects classified as belonging to the minority class.

$$fallout(\Psi, \mathcal{VS}) = \frac{FP}{FP + TN} \tag{2.37}$$

Precision (also known as positive predictive value) [190], informing about the model's ability to correctly detect minority class objects. Indicates how many of the objects assigned by the model to the positive class actually belongs to said class.

$$precision(\Psi, \mathcal{VS}) = \frac{TP}{TP + FP} \tag{2.38}$$

Balanced accuracy score (BAC) [34, 120], defined for multi-class problems as the average of *recall* calculated on each class. For binary problems, it is the average of *recall* and *specificity*.

$$BAC(\Psi, \mathcal{VS}) = \frac{Recall + Specificity}{2} \tag{2.39}$$

Geometric mean score [11, 147], known in two versions. By far the most popular is defined as the square root of the product of *recall* and *specificity* ($Gmean_s$). However, there is also an alternative definition where *specificity* is replaced by *precision* ($Gmean$).

$$Gmean_s = \sqrt{Recall * Specificity} \tag{2.40}$$

$$Gmean = \sqrt{Recall * Precision} \tag{2.41}$$

F_β *score* [9], which is interpreted as the weighted harmonic mean of *recall* and *precision*. Thanks to this, it takes into account both of these base metrics, while punishing extremely low values of either of them. The β parameter expresses how many times *recall* is more important than *precision* and can be tuned, resulting in different trade-offs between both metrics. Using this metric could be dangerous if the parameter is not set properly. Brzezinski et al. [41] show, that unsuitable β value may resulting in favoring the majority class. F_β *score* is also criticized due to asymmetric response to the dynamically changing *Imbalance Ratio* and being more susceptible to simple oversampling [36].

$$F_\beta = (1+\beta^2) * \frac{Precision * Recall}{(\beta^2 * Precision) + Recall} \tag{2.42}$$

F_1 *score* [205] can be interpreted as F_β *score*, where the β value is 1. It is defined as the harmonic mean of *recall* and *precision*.

$$F_1\ score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.43}$$

Another way to evaluate to classifiers performance is to use two graphical-based metrics [29], namely *Receiver Operating Characteristics* (ROC) curve and the corresponding area under the ROC curve (AUC) [227]. The ROC curve allows the visualization of trade-off between the FPR (x axis) and TPR (y axis) for given value of threshold used for labeling a sample as belonging to the positive class. The point $(0, 1)$ represents a perfect classifier, the point $(0, 0)$ is a classifier that predicts all samples as negative, $(1, 1)$ is the classifier that labels all samples as belonging to the positive class, and the point $(1, 0)$ is the classifier which is always incorrect. The ROC curve has been widely used in the case, where the classification cost is hard to obtain. AUC allows the models comparison or general evaluation of a single classifier, averaged over different parameter values [78]. Nevertheless, Hand deemed AUC as fundamentally incoherent and proposed the alternative measure [101].

Static imbalanced data sets

Chapter 3 of this dissertation focuses on the classification of static imbalanced data. Table 2.3 presents the characteristics of 41 datasets selected from the KEEL [4] repository. All datasets have a high imbalance ratio of at least 9 and contain binary problems that were generated through various combinations of class merging.

Experimental protocol

In this thesis, all experiments on static data sets will be conducted according to the k-fold cross-validation evaluation protocol [124]. In this approach, a dataset is randomly divided into k mutually exclusive folds of equal size. Then, $k-1$ folds are used for training the algorithms and the remaining one for evaluation. This procedure is repeated until

Table 2.3: *Imbalanced datasets characteristics.*

Dataset	#I	#F	IR	Dataset	#I	#F	IR
ecoli-0-1_vs_2-3-5	244	7	9	glass2	214	9	12
ecoli-0-1_vs_5	240	6	11	glass4	214	9	15
ecoli-0-1-3-7_vs_2-6	281	7	39	glass5	214	9	23
ecoli-0-1-4-6_vs_5	280	6	13	led7digit-0-2-4-5-6-7-8-9_vs_1	443	7	11
ecoli-0-1-4-7_vs_2-3-5-6	336	7	11	page-blocks-1-3_vs_4	472	10	16
ecoli-0-1-4-7_vs_5-6	332	6	12	shuttle-c0-vs-c4	1829	9	14
ecoli-0-2-3-4_vs_5	202	7	9	shuttle-c2-vs-c4	129	9	20
ecoli-0-2-6-7_vs_3-5	224	7	9	vowel0	988	13	10
ecoli-0-3-4_vs_5	200	7	9	yeast-0-2-5-6_vs_3-7-8-9	1004	8	9
ecoli-0-3-4-6_vs_5	205	7	9	yeast-0-2-5-7-9_vs_3-6-8	1004	8	9
ecoli-0-3-4-7_vs_5-6	257	7	9	yeast-0-3-5-9_vs_7-8	506	8	9
ecoli-0-4-6_vs_5	203	6	9	yeast-0-5-6-7-9_vs_4	528	8	9
ecoli-0-6-7_vs_3-5	222	7	9	yeast-1_vs_7	459	7	14
ecoli-0-6-7_vs_5	220	6	10	yeast-1-2-8-9_vs_7	947	8	31
ecoli4	336	7	16	yeast-1-4-5-8_vs_7	693	8	22
glass-0-1-4-6_vs_2	205	9	11	yeast-2_vs_4	514	8	9
glass-0-1-5_vs_2	172	9	9	yeast-2_vs_8	482	8	23
glass-0-1-6_vs_2	192	9	10	yeast4	1484	8	28
glass-0-1-6_vs_5	184	9	19	yeast5	1484	8	33
glass-0-4_vs_5	92	9	9	yeast6	1484	8	41
glass-0-6_vs_5	108	9	11				

the chosen metric is estimated based on all available folds, i.e., k times. The final metric values is calculated as the average of k metric estimations. The whole process can also be repeated a set number of times, resulting in repeated cross-validation protocol. The value of the parameter k usually depends on the dataset size, i.e., the more problem samples, the smaller the k. Recommended values are $k = 10$ or $k = 5$. As the random splitting may lead to so-called *dataset shift*, in which the folds obtained are not representative of the original dataset, the protocols based on *stratified sampling* have been proposed [176]. One such approach is the *standard stratified cross-validation* which maintains in each fold the original class distributions and will be used in the following thesis.

The use of cross-validation allows, apart from desensitizing to the random factor, for performing the null hypothesis statistical tests [215]. Such tests enable answering the question, whether the obtained performance difference is statistically significant. Stapor et al. describe three scenarios, in which the statistical tests can be applied [219]:

- Two classifiers – one dataset,
- Two classifiers – multiple datasets,

- Multiple classifiers – multiple datasets.

For the comparison of two classifiers on one dataset, when using the repeated cross-validation protocol, the most popular are the classical *t test* and the corrected *t test* [27]. When comparing two classifier on multiple datasets, the Wilcoxon signed-rank test is widely recommended [111]. For the comparison of multiple classifiers on multiple datasets, the recommended methodology is to first use the omnibus test in order to check if any model differs from other. The most popular omnibus test is the Friedman non-parametric test [111]. In the second step, if the null hypothesis of the omnibus test is rejected, the *post-hoc analysis* with multiple hypothesis testing is performed, which for Friedman test in based on the means ranks.

2.4.2 Data stream

The main characteristic of the data stream classification [135] is the possibility of the large amount of data appearing sequentially, creating endless data stream over which the observer has no influence when it comes to the order at which instances arrive. Moreover, a classifier has to be ready at all times to make a decision. When designing effective classifier for data streams, we have to consider a few important issues:

- Possibility of changes in data distribution (*concept drift*),
- Frequent need for quick classifying of incoming samples,
- Delay or impossibility of data labeling,
- Limited resources as memory, storage, and computational power.

For the purposes of the following thesis, the data stream is defined as a set of data chunks $\mathcal{DS}_k$ with fixed-size N, where k is the chunk index. and Ψ_k denotes the classifier trained based on the kth chunk.

Because not all objects can be stored in memory, it is widely accepted that each instance may be processed at most one time, and it is not remembered. Therefore its re-evaluation could be impossible. Usually, information about instances is replaced by statistics. Finally, we may be faced with non-stationary data streams, i.e., where parameters of the classification model (characteristics of probabilistic distributions) may change, forcing the classification model to adapt to upcoming changes. This phenomenon is called *concept drift* and its nature can vary due to both the character and the rapidity. It forces the implementation of mechanisms enabling adapting to the current class imbalance status or concept drift detectors that providing a drift occurs enforces the model to be rebuilt. From the classification point of view, we can distinguish two types of such an event: (*i*) the real concept drift that can strongly affect the shape of the decision boundary; and (*ii*) the virtual drift that does not affect the decision rule. Another drift taxonomy depends on the drift impetuosity:

- slow changes, i.e., *incremental drift.*
- abrupt changes, i.e., *incremental drift.*

It is difficult to assign a *gradual drift* in this taxonomy. On the one hand, it can be considered as a slow-moving change, but on the other hand, it can be seen as an abrupt change related to class overlapping.

Additionally, we can consider a reoccurring concept drift. It may occur in cases of, e.g., seasonal phenomena as weather prediction or client preferences of clothes or sports stores. It is worth emphasizing that the presence of a *concept drift* can lead to serious deterioration of the classifier's accuracy. Therefore, developing efficient methods that are able to deal with this type of change in the data stream is nowadays the focus of intense research.

Kuncheva analyzed various approaches to streaming data classification employing classifier ensemble techniques in [149]. Based on this analysis, the following strategies can be distinguished:

- Dynamic combiners, where the classifier ensemble changes the rule by which trained in advance base models are combined (e.g., changing weights for weighted voting) [161],
- Updating training data, where base classifiers are updated in an online manner using incoming training instances, (e.g., in online bagging [185] or leveraging bagging [20]),
- Updating base classifiers [126],
- Updating the classifier ensemble line-up, where, e.g., the oldest or worst performing classifier is replaced by a new one, trained on the most recent data [256].

Based on the approach to data processing, classifiers dedicated to the data stream classification task can be categorized into chunk/batch-based or online methods. Batch-based methods process the stream in chunks, which contain a fixed number of samples. This allows iterating several times of samples in each chunk to generate base classifiers. Online learning methods process each sample individually after its arrival, which is an approach dedicated for scenarios with strict memory and time constraints [135].

Despite the large number of methods proposed, the classifier ensemble remains the focus of intense research and is one of the more promising directions of the data stream analysis, both stationary and non-stationary. Still, constructing a well-performing ensemble of classifiers is strongly related to the method of ensuring high diversity of the classifier pool and employed combination method [253].

One the most well recognized ensemble approaches to stationary data stream classification is the *Learn++* algorithm proposed by Polikar et al. [189]. *Learn++* trains a neural network model on each incoming chunk and adds it to the pool, which is combined using *majority voting*. All models are retained in the pool. Zhao et al. proposed *Bagging++* [269] as an improvement for *Learn++*. This approach employs *Bagging* to generate new models from each data chunk, using four different learning algorithms. Minku et al. introduced the *Growling Negative Correlation Learning Growling* NCL [172] algorithm, aimed at co-training a classifier ensemble composed of diverse and accurate neural networks.

Online ensembles for stationary data stream classification include *Online Bagging* OB, proposed by Oza [184], which uses the *Poisson(*$\lambda = 1$*)* distribution to update each base classifier with the appearance of a new instance. Bifet et al. two algorithms modifying OB, namely *Adaptive-Size Hoeffding Trees* (ASHT) [21] and Leveraging Bagging (*LevBag*) [20]. Both of those methods aimed at randomizing the classifiers' input and ouput. ASHT does that by generating decision trees of different sizes, while *LevBag* allows specifying the value of λ parameter during resampling and employs output detection codes. Another approach proposed by Oza is the *Online Boosting* (*OzaBoost*) [184]. Here, a fixed-size ensemble is maintained and the classifiers are sequentially updated using each incoming. The weights of misclassified instances are increased in order to emphasize them when updating models. Gama proposed *Hoeffding Option Trees* (HOT) ensemble [88], which allows updating a set of option nodes instead of a single leaf.

One of the most well known example of batch-based classifier ensemble algorithm for the non-stationary data stream classification task is the *Streaming Ensemble Algorithm* (SEA) [221] proposed by Street and Kim, which trains a new base model on each incoming data chunk and adds it to the classifier pool but removes the worst model if the pool size is exceeded. Wang et al. introduced the *Accuracy Weighted Ensemble* (AWE algorithm [239], which is a standard ensemble batch processing method based on *mean square error* calculations. Brzezinski and Stefanowski proposed the extension od AWE, namely the *Accuracy Updated Ensemble* (AUE) algorithm [38] allowing for updating the member classifiers. The *Weighted Aging Ensemble* (WAE) [256], modifying AWE by changing the weights calculation and classifier selection methods, was proposed by Wozniak et al. Elwell and Polikar *Learn++ for non-stationary environments Learn++.*NSE [77], inspired by *Learn++*, sets the weights of training samples from each chunk based on the error obtained when classifying it.

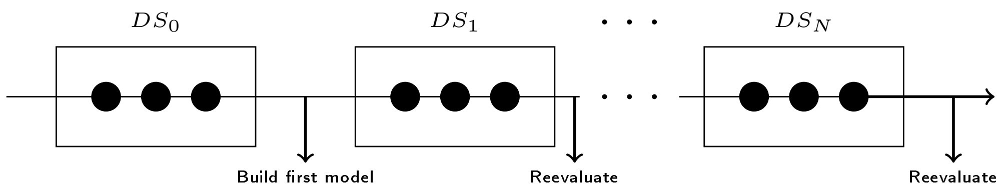
Regarding online ensemble methods for non-stationary data stream classificatio, one of the popular approaches is the *Dynamic Weighted Majority* (DWM) proposed by Kolter and Maloof [127]. In DWM, each base classifier has a weight, which is reduced each time a wrong prediction is made. Brzezinski and Stefanowski introduced the incremental version of AUE, namely *Online Accuracy Updated Ensemble* (OAUE) [39], which employs the new cost-effective function for classifier weighting. Yoshida et al. proposed the WWH algorithm [264], which combines an adaptive ensemble with instance selection based on overlapping windows. The *Sparse Online Classification* (SOC) framework from Wag et al. [238] uses sparse online learning algorithms for online drifting data stream classification.

Data stream classifier evaluation

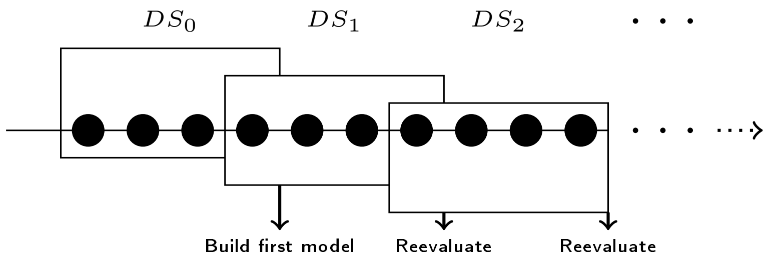
As previously mentioned, cross-validation is the most commonly used evaluation approach in learning from static data. However, in the case of learning from the data stream, this method cannot be used due to, among other factors, computational limitations due to potentially huge amounts of data, as well as possible concept drift or dynamic imbalance occurrence [135].

Concerning batch data stream analysis, which is one of the main topics of interest in the following thesis, two approaches are often employed:

- Test-Then-Train [88], shown in Figure 2.3 a. Each individual chunk is first used to test the current classification model and then to update it. The first data chunk in a data stream is used to initialize the classification model, skipping the prediction step.
- Prequential [89] (Figure 2.3 b), which in the batch-based version relies on the forgetting mechanism in the form of a sliding window, rather than on separate data chunks. After each prediction and update step, the window moves by a fixed number of instances, keeping some of those previously seen. This makes the approach more sensitive to changes occurring in the stream. However, it is associated with an increase in the computational cost. An example of an evaluation based on this technique can be the prequential AUC proposed by Brzeziński and Stefanowski for imbalanced data stream classification [40].



(a) *Test-Then-Train*



(b) *Prequential*

Figure 2.3: *Data stream evaluation schemes.*

There also exist other approaches for comparing data stream classification methods. An example are metrics for assessing the behavior of classification methods during a concept drift occurrence, proposed by Shaker and Hüllermeier [210], namely the *restoration time* and the *maximum performance loss*. Let's denote two stationary streams generated according to distributions P_A and P_B as DS_A and DS_B. The drifting data stream generated by random sampling of DS_A and DS_B is defined as DS_C.

Restoration time informs about the length of the algorithm's recovery phase after the *concept drift* occurrence, and is defined as

$$\frac{t_2 - t_1}{T} \in [0, 1], \tag{2.44}$$

where t_1 is the time at which the learning curve DS_C drops below 95% of the performance curve DS_A, t_2 is the time at which the learning curve DS_C recovers up to 95% of the performance curve DS_B, and T denotes the length of the entire data stream.

The *maximum performance loss* measures the maximal decrease in the method performance in the event of concept drift. In classification task, it compares DS_C with the pointwise minimum

$$DS(t) = \min\{DS_A(t), DS_B(t)\} \tag{2.45}$$

as a baseline and computes the *maximum performance loss* as compared to this baseline

$$\max_{t \in T} \frac{DS(t) - DS_C(t)}{DS(t)}. \tag{2.46}$$

Another problem in the case of data stream classification is the method of carrying out the statistical analysis of the obtained results. So far, there have been few solution proposals for this issue [18]. One such approach is to perform standard statistical tests using metric values averaged over the entire length of the data stream - which requires the use of synthetic streams to generate a replication of the said stream with the same characteristics but a different random seed. This approach, however, tries to transform a dynamic problem into a static one and does not take into account the changes occurring during the entire length of the evaluation process. This approach is also not applicable to real data streams. Another method is to use a sliding window or separate data chunks. However, due to a large number of degrees of freedom, the results are almost always statistically independent of each other. For this reason, there is currently no defined approach to performing statistical tests on single data streams.

2.4.3 Imbalanced data stream

Despite the fact that real-life data streams may often display a high degree of imbalance, there is still a scarcity of articles trying to combine both non-stationary data stream and imbalanced data classification tasks [37]. Additionally, it is often overlooked that imbalanced data streams may be characterized by the dynamic changes in the *Imbalance Ratio*, which may be regarded as the equivalent of *concept drift* phenomenon for *prior* class probabilities. The analysis of literature in the field of non-stationary data stream shows that the vast majority of works deal with problems of changes in the posterior probability, relatively rarely addressing the topic of imbalanced streams, and in particular, dynamically imbalanced streams, i.e. those characterized by changes in the prior probability [243].

Existing methods for mining imbalanced data streams, same as for balanced ones, work in two distinctive modes, i.e., the data is arriving in chunks and data windows are given for processing or the data is processed online. Work by Gao et al. [90] is worth highlighting as a technique based on the notion of classifier ensemble, where each of the individual learners is generated using instances from the majority class in the consecutive data windows as well as on the already accumulated minority class instances. In [246], authors propose an ensemble approach, where before learning on each upcoming data windows *undersampling* is performed based on the *k-Means* algorithm. Chen et al. [53] follow the same technique and describe a family of algorithms SERA, MuSeRA and REA, which add selected from the appearing minority class objects to the currently processed data window. In [160], authors discuss a method for calculating the weights of classifiers learned on data windows and using combination rule based on weighted voting. In [72] authors propose a modification of the *Learn++* algorithm for imbalanced data (*Learn++*.NIE and *Learn++*.CDS). Both methods, while achieving good recognition ability, require significant computational resources. An interesting approach, also employing classifier ensembles, in which the *Imbalance Ratio* dynamically changes were proposed by Sun et al. [223]. The second group of methods are based on incremental (online) learning mode. Nguyen et al. described an approach based on *Random Oversampling* [181], while in [245], authors propose an interesting method called *Sampling-based Online Bagging*, employing both *undersampling* and *oversampling*. The decision on which model to use at the given time is made based on the outputs of both *imbalance ratio detector* and *drift detector*. Worth mentioning is also the work on the RLSACP by Ghazikhani et al. [94], and WOS-ELC algorithm by Zong et al. [275]. The aim of these methods is to set the perceptron weights in a way preferring the minority class.

Real data streams

Unfortunately, when it comes to the task of classifying imbalanced data streams with concept drift, there are many limitations in accessing the real data. There are some works that present an overview of the databases available for this type of problem [47, 63, 164]. Alas, after discussion with some of the authors of these articles and thoroughly checking the data streams they listed, the use of provided data streams for this particular problem turned out to be difficult.

That was due to various factors, such as:

- The problem turned out to be too simple,
- The stream contained instances appearing sequentially in classes,
- The data stream did not have noticeable or definable concept drifts,
- The data did not have an appropriate imbalance ratio.

Some of these problems could be addressed by modifying the actual data stream (e.g., by reshuffling or injecting drift). However, this approach was not used, as such a solution would destroy the actual data structure and would amount to researching artificially generated data. Due to the low availability of data that would allow reliable verification of the proposed algorithms in terms of their behavior when classifying imbalanced data streams with concept drift, based on preliminary study, five benchmark streams were selected. All streams were binarized artificially (by combining classes). Both *covtypeNorm-1-2vsAll* and *poker-lsn-1-2vsAll* [47] do not have a definable type of drift. Also, in order to make them usable during experiments, the longest possible section intervals were selected from both streams. This was done in order to guarantee the appearance of samples from both classes in each chunk containing 1000 instances. In the case of INSECTS data [218], the streams have distinct - predefined - types of concept drift. However, to make things more difficult, a tool included in the data stream mining framework MOA [19] was used to establish the minority class size in each of these three problems at 5%.

The characteristics of selected real data streams are presented in Table 2.4.

Table 2.4: *Real data streams characteristics.*

Data stream	**#Samples**	**#Features**	**IR**
covtypeNorm-1-2vsAll	266 000	54	4
poker-lsn-1-2vsAll	360 000	10	10
INSECTS-abrupt_imbalanced_norm	300 000	33	19
INSECTS-gradual_imbalanced_norm	100 000	33	19
INSECTS-incremental_imbalanced_norm	380 000	33	19

Synthetic data streams

Based on the above-mentioned conclusions, it was considered necessary to use synthetic data stream generators to evaluate the methods proposed in the thesis. Thanks to this, the behavior of algorithms under strictly defined conditions can be tested. The variety of streams can be ensured by generating a number of replications, based on the determined random seeds, for each combination of parameters such as: (*i*) the *Imbalance Ratio*, (*ii*) the level of label noise, defining the global percentage of incorrect labels occurrence, and (*iii*) the type of *concept drift*.

One of the commonly used generators are those available in the MOA data stream mining framework [19]. Aside from the above-mentioned parameters, these streams differ in the generator used and the number of attributes. The following generators are important in the context of the following dissertation: (*i*) *Agrawal - sudden* and *gradual concept drift*, 9 attributes, (*ii*) *Hyperplane - incremental concept drift*, 10 attributes.

The vast majority of research presented in the following thesis has been carried out on synthetic data streams generated using stream-learn package for difficult data stream batch analysis [141], developed in collaboration with Dr Paweł Ksieniewicz.

2.4.4 Partially labeled data

Another critical problem encountered during streaming data analysis is access to the correct label for incoming objects. Many of the methods described in the literature ignore this topic, assuming that labels are always available. They ignore the fact that even if the labels for the incoming objects can be obtained, samples can arrive fast enough, that labeling all of them will be impossible. The cost of labeling should be also taken into consideration. Sometimes this cost is negligible, e.g., in the case of weather forecasting (a label can be obtained with a delay, but the cost is only related to the observation and imputing it into the system). However, in most cases, such as medical diagnostics, labels are the result of human experts' effort, so labeling involves the cost of their work. Given the above, the assumption that labels can be obtained for free is unrealistic and limits the possibility of using many methods in real-life decision problems [1].

The following thesis deals partially with minimizing the necessary cost of data labeling using the so-called *active learning* approach [207]. It concentrates on choosing the *interesting* unlabeled objects, which are then passed as queries to be labeled by the expert. There three main *active learning* scenarios that have been considered in the literature:

- *Membership Query Synthesis* [7] – In this scenario, the learner can request labels for any unlabeled samples, but typically the queries relate to the instances synthesized by the learner. The labeling of the generated instances can be problematic if the annotator is a human expert. For example, in image classification, the generated instances may not contain meaningful objects [14]. However, this scenario shows promising results when the labels are not derived from a human annotator but are, for example, the result of experimentation [121].
- *Stream-Based Selective Sampling* [8, 55] – The assumption of this scenario is that obtaining an unlabeled sample is inexpensive (or basically free). Because of that, an instance can be first samples from the distribution, and then the learner can decide whether it wants to query an expert about its label. Samples are evaluated based on various *query strategies* [64], like e.g., *uncertainty sampling* [158], *Query-By-Committee* (QBC) [209], or *Expected Gradient Length* (EGL) [208].
- *Pool-Based Sampling* [158] – This scenario is motivated by the fact, that for many real-world tasks, a large collections of unlabeled samples can be gathered simultaneously. In contrast to the *stream-based selective sampling*, query decisions are not made individually in a sequential manner, but the collection of samples is evaluated and ranked before selecting the best query.

The use of *active learning* for streaming data processing has been noticed, among others [152, 274], however it is still not widely used. Hence, it is worth noting the work of Bouguelia et al. [28], who proposed a new *active learning query strategy* based on instance weighting. Ksieniewicz et al. [145] used *query by example* based on the values of the *support function* to improve neural network's prediction. [136] proposed employing different (*query by committee*) to classify non-stationary data stream. It is also worth mentioning the work [211], where the authors build a classifiers ensemble employing both the *active learning* approach as well as random labeling. Yu et al. proposed the *extreme learning machine* based solution, called *Active Online-Weighted Extreme Learning Machine* AOW-ELM [265]. A hybrid labeling strategy based on *uncertainty sampling* and class distribution was proposed for the imbalanced data stream classification by Zhang et al. [268].

Another approach aiming to deal with the problem of limited access to labels is known as *self-labeling* [273]. The goal of these techniques is to enlarge the original learning set (or obtain several extended learning sets) by adding unlabeled samples with the most confident predictions.vIn the literature, *Self labeling* is usually divided into (i) *self-training* [159, 262], where classifier is trained using small initial pool of labeled samples and then retrained using learning set extended by its most confident predictions, and (ii) *co-training* [2, 25], which assumes that the feature space can be split into two independent sets called *views*. Then one classifier is trained on each *view* and they teach each other the most confident predictions. Triguero et al. defined the main properties of *self-labeled* techniques [230]:

- *Addition mechanism*, which defines whether an enlarged labeled set is obtained incrementally, in batch mode, or by amending.
- *Single-classifier versus multi-classifier*, which specifies how many classifier are used during the enlarging process.
- *Single-learning versus multi-learning*, which defines whether the used classifiers are heterogeneous of homogeneous.
- *Single-view versus multi-view*, which specifies how the feature space is considered by the *self-labeled* algorithm.

The example of employing *self-labeling* in the data stream classification task might be the *Scaffolding Type-2 Classifier* proposed by Pratama et al.[191] and (*ST2Class*) based on *Fuzzy Neural Network*. Korycki et al. augmented the *active learning* module using *self-labeling* in order to improve data stream classification under very small instance budget [129].

2.5 Stream-learn library for difficult data stream batch analysis

The *stream-learn* is a *Python* module, implementing the *scikit-learn* API [187], intended for a batch-oriented data stream processing. It implements a data stream generator, based on the *Madelon* [99] model used to generate static data in *scikit-learn* and allows the development of both stationary and dynamic data streams, containing both *concept* and *prior class probabilities drifts*. It is supplemented with exemplary, simple stream classifiers (*Accumulated Samples Classifier* and *Sample Weighted Meta Estimator*), which may be used as the boilerplate for the users' solutions, and *state-of-art* classifier ensembles (SEA (*Streaming Ensemble Algorithm*) [221], *OnlineBagging* [184], OOB (*Oversampling-Based Online Bagging*) [243], UOB (*Undersampling-Based Online Bagging*) [243], AWE (*Accuracy Weighted Ensemble*) [239], AUE (*Accuracy Updated Ensemble*) [38] and WAE (*Weighted Aging Ensemble*) [256]). The package also implements evaluation metrics that are more computationally effective than those available in *scikit-learn* and *imbalanced-learn*. The element wrapping-up the package and allowing for conducting experiments is a pair of evaluators: *Test-Then-Train* [88] and *Prequential* [89], in their batch variants.

Software Architecture

The *stream-learn* package is organised in five modules, responsible for (*i*) data streams, (*ii*) evaluation methods, (*iii*) classification algorithms, (*iv*) classifier ensembles and (*v*) evaluation metrics. A general diagram of the project architecture is shown in Figure 2.4.

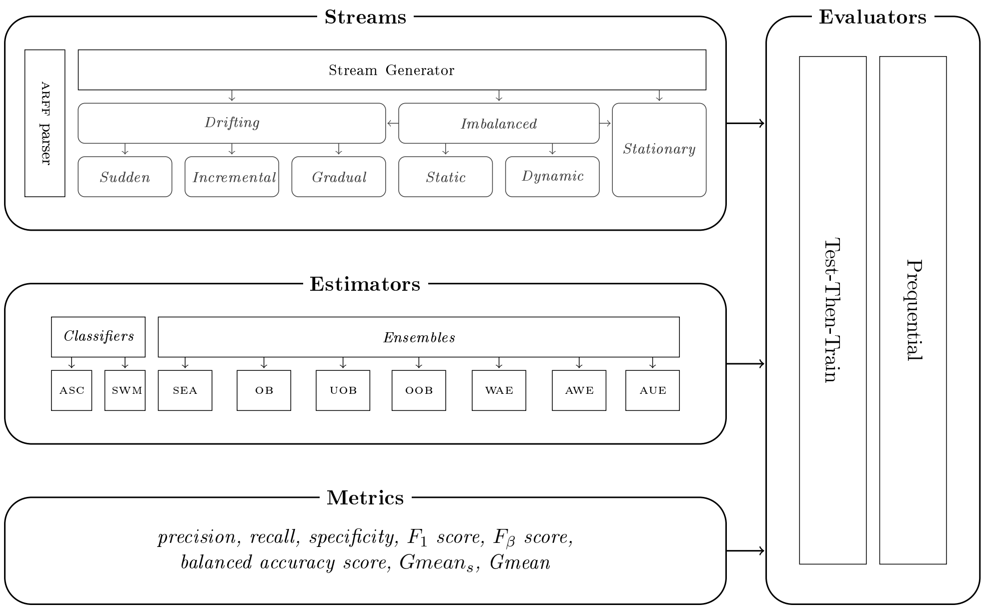


Figure 2.4: *Overall schema of the software architecture.*

The *streams* module contains the ARFF file parser class, which is the standard format for serialising both real data streams and those generated, for example, by the MOA software, as well as the *StreamGenerator* class responsible for generating synthetic data streams. A more detailed description of the module can be found in Section 3.

The *evaluators* module contains classes responsible for two main prediction measures estimation techniques on data streams, namely *Test-Then-Train* and *Prequential*, in their batch-based versions. The former one is based on separate windows known as data chunks, while the latter uses a sliding window as a forgetting mechanism. Both techniques, in each step, reevaluate existing classifiers.

Estimators can be found in the *classifiers* and *ensembles* modules, which contain the classifiers adapted for stream classification and *state-of-art* classifier ensembles that can be used with implemented estimators.

The module *metrics* implements a variety of evaluation measures for unbalanced binary classification [42]. The decision to create a new implementation was made due to the low computational efficiency of the metrics included in existing packages. The module includes *recall* [190], *precision* [190], F_β *score* [9], F_1 *score* [205], BAC [34, 120], $Gmean_s$, and *Gmean* [11, 147].

Data stream generation A key element of the *stream-learn* package is a generator that allows a replicable (according to the given *seed*) classification dataset to be created with a class distribution that changes over the course of a data stream, with basic concepts built on a standard class distribution for the *scikit-learn* package from the *make_classification()* function. These types of distributions attempt to reproduce the rules for generating the *Madelon* set [99]. The *StreamGenerator* is capable of generating any variant of the stream known in the general taxonomy of streams.

Stationary Stream The simplest types of data streams are *stationary streams.* They contain a basic concept that is static for the entire course of processing. The chunks differ from each other in terms of the patterns they contain, but the decision boundaries of the models built on them should not differ statistically. This type of stream can be generated with a clean generator call with no additional parameters. Such a stream is shown in Figure 2.5, which contains the set of *scatter plots* for a two-dimensional stationary stream with the binary problem.

What is important, contrary to a typical call to *make_classification()*, the *n_samples* parameter, determining the number of patterns in the set, is not specified here, but instead, two new attributes of a data stream are provided:

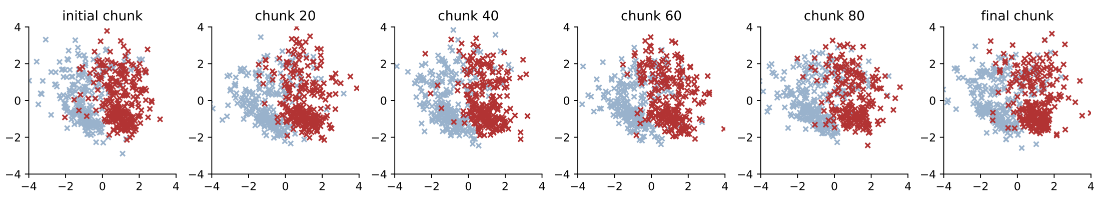


Figure 2.5: *Scatter plots of selected chunks from a* stationary data stream.

- *n_ chunks* — to determine the number of chunks in a data stream.
- *chunk_ size* — to determine the number of patterns in each data chunk.

In addition, data streams may contain noise, which is not considered *concept drift* but presents an additional challenge during stream analysis and against which classifiers should be robust. The *StreamGenerator* class implements noise by inverting the class labels of a certain percentage of the incoming instances in the data stream. This percentage can be defined by an *y_flip* parameter, as in the standard *scikit-learn* dataset generation call. If a single float is specified as the parameter value, the percentage of noise refers to combined instances from all classes. On the other hand, if a tuple of floats is specified, the noise is done separately within each class using the specified percentages.

Data streams containing concept drift The most commonly studied property of data streams is their variability over time. The phenomenon of *concept drift* is responsible for this. The *stream-learn* package attempts to address the need to synthesize all the basic variants of this phenomenon (i.e., *sudden*, *gradual*, and *incremental drifts*).

Sudden (Abrupt) drift

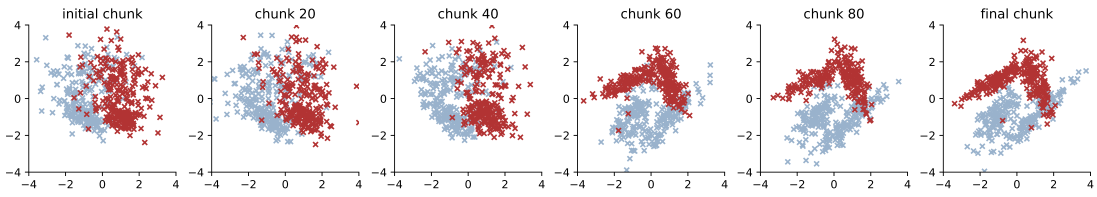
This type of drift occurs when the concept from which the stream is generated is suddenly replaced by another. The concept probabilities used by the *StreamGenerator* class are created based on a sigmoid function generated with the *concept_ sigmoid_ spacing* parameter, which determines the shape of the function and the suddenness of the concept change. The higher the value, the more sudden the shift. Here, this parameter takes the default value of 999, which allows for the generation of a sigmoid function that simulates an abrupt change in the data stream. An illustration of sudden drift is shown in Figure 5.1.

Gradual drift

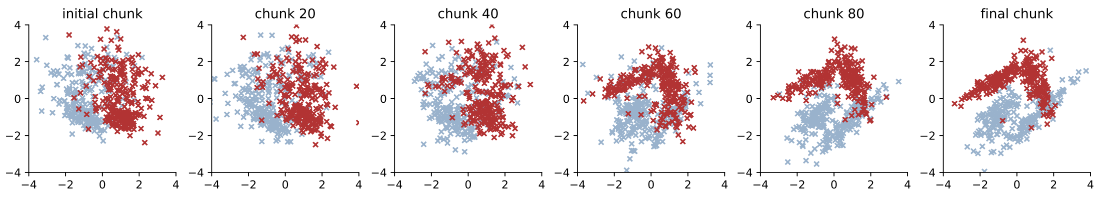
Unlike *sudden drifts*, gradual drifts are associated with a slower rate of change that can be detected by observing the data stream for a longer period of time. This type of drift refers to the transition phase in which the probability of obtaining instances of the first concept decreases, while the probability of obtaining instances of the next concept increases. The *StreamGenerator* class simulates *gradual drift* by comparing the concept probabilities with the generated random noise and selecting which concept is active at a given time depending on the result. An illustration of *gradual drift* is shown in Figure 5.2.

Incremental (gradual) drift

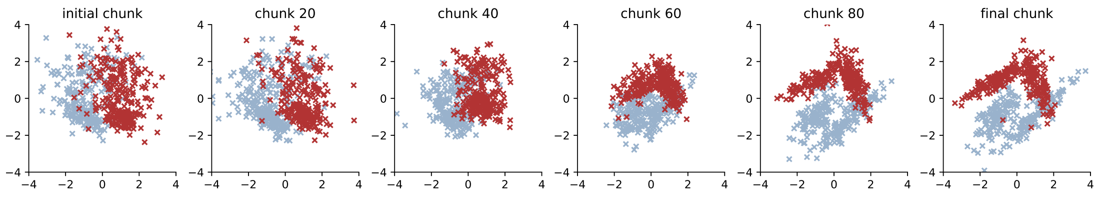
incremental drift occurs when a series of barely perceptible changes in the concept used to generate the data stream occur, unlike *gradual drift* where samples from different concepts are mixed without changing. For this reason, drift can only be detected after some time. The severity of the changes, and thus the speed of transition from one concept to another, is described by the parameter *concept_ sigmoid_ spacing*, as in the previous example. An illustration of *incremental drift* is shown in Figure 5.3.



(a) *Data stream with sudden drift.*



(b) *Data stream with gradual drift.*



(c) *Data stream with incremental drift.*

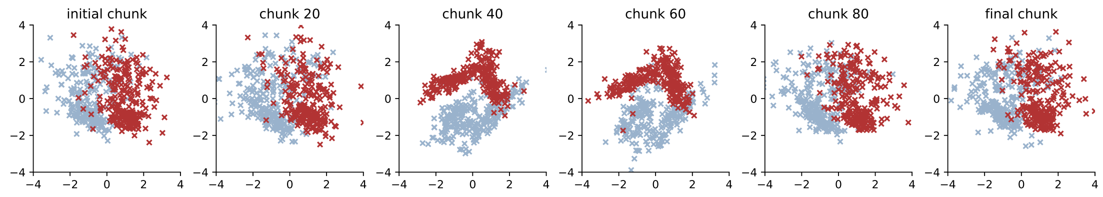
Figure 2.6: *Changes in class distribution under each type of concept drift.*

Recurrent drift

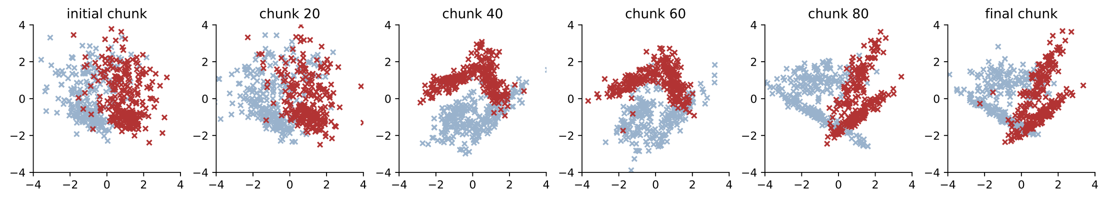
The cyclic repetition of class distributions is an entirely different property of concept drifts. If after another drift, the concept earlier present in the stream returns, we are dealing with a *recurrent drift.* We can get this kind of data stream by setting the *recurring* flag in the generator. Illustration of the *recurrent drift* is presented in Figure 2.7a.

Non-recurring drift

The default mode of consecutive concept occurrences is a non-recurring drift, where in each concept drift an entirely different new, previously unseen class distribution is synthesised. Illustration of the *non-recurring drift* is presented in Figure 2.7b.



(a) *Data stream with recurring drift.*



(b) *Data stream with non-recurring drift.*

Figure 2.7: *Changes in class distribution under recurring and non-recurring concept drift.*

Class imbalance

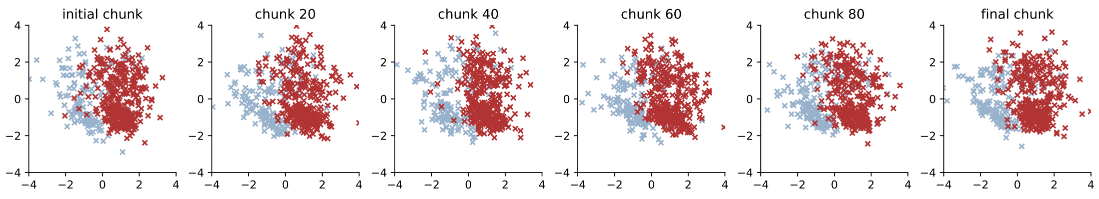
Another area of data stream properties, different from a *concept drift* phenomenon, is the prior probability of problem classes. By default, a balanced stream is generated, i.e. one in which patterns of all classes are present in a similar number.

Stationary imbalanced stream The primary type of problem in which we are dealing with disturbed class distribution is a *stationary imbalanced stream*, where the classes maintain a predetermined proportion in each chunk of a data stream. To acquire this type of a stream, one should pass the *list* to the *weights* parameter of the generator (*i*) consisting of as many elements as the classes in the problem and (*ii*) adding up to one. Illustration of the *stationary imbalanced* stream is presented in Figure 2.8a.

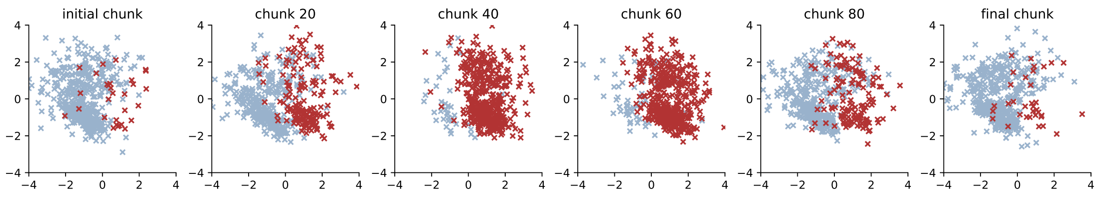
Dynamically imbalanced stream A less common type of *imbalanced data*, impossible to obtain in static datasets, is *data imbalanced dynamically.* In this case, the class distribution is not constant throughout a stream, but changes over time, similar to changing the concept presence in gradual streams. A *tuple* of three numeric values is passed to the *weights* parameter of the generator to get this type of a data stream:

- the number of cycles of distribution changes.
- *concept_sigmoid_spacing* parameter, deciding about the dynamics of changes on the same principle as in *gradual* and *incremental drifts.*
- a range within which oscillation is to take place.

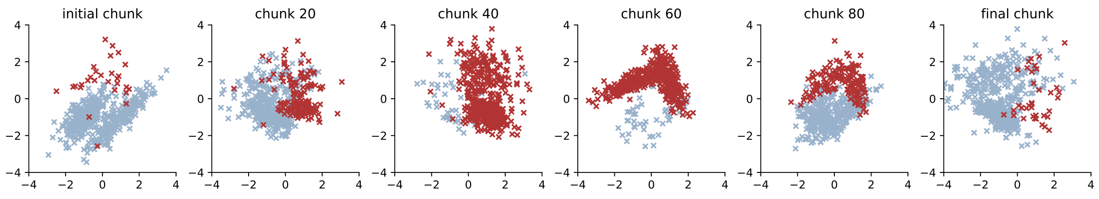
Illustration of the *dynamically imbalanced stream* is presented in the Figure 2.8b.



(a) *Statically imbalanced data stream.*



(b) *Dynamically imbalanced data stream.*



(c) *Dynamically Imbalanced Stream with Concept Oscillation (DISCO).*

Figure 2.8: *Changes in class distribution under dynamically changing prior class probabilities (a,b) and concept drift paired with dynamic imbalance (c).*

Mixing drift properties When generating data streams, we do not have to limit ourselves to just one modification of their properties. One may easily prepare a stream with many drifts, any dynamics of changes, a selected type of drift and a diverse, dynamic imbalanced ratio. The last example of a data stream is such a proposition, namely, DISCO (*Dynamically Imbalanced Stream with Concept Oscillation*). Illustration of the DISCO stream is presented in Figure 2.8c.

Impact The articles conducted so far using *stream-learn* package deal with application of preprocessing in the incremental imbalanced data stream classification methods [98], active learning techniques [145] and exploring the possibilities of employing the *Dynamic Ensemble Selection* [277, 280, 284]. Thanks to the precise, replicable and user-friendly stream generation procedure, it also allows for a broad spectrum of *drift detection* analyses, depending not only on types of drifts but also on the dynamics of their changes. Finally, it also implements online bagging methods (UOB, OOB), which, to the knowledge of the authors, have not yet had open and stable implementation. Additionally, thanks to the implementation of the ARFF files parser, the *stream-learn* allows for convenient work with real data streams, which may help to solve actual problems in real-life scenarios.

Chapter 3

Algorithms for imbalanced data classification

In this chapter, methods dedicated to the task of difficult stationary data classification will be presented. Ensemble methods remain one of the leading approaches in the difficult data classification problem. Therefore, there is a need to introduce new classifier selection methods, as well as new approaches to classifier combination.

First, three methods focusing on clustering-based ensemble pruning are presented. These types of approaches look for the group of similar classifiers which are replaced by their representatives. A novel pruning criterion, based on well-known diversity measures, is proposed. The first method selects the model with the best predictive performance from each cluster to form the final ensemble, the second one employs the multistage organization, where instead of removing the classifiers from the ensemble each classifier cluster makes the decision independently, while the third proposition combines multistage organization and sampling with replacement. Next, two methods, using the similarity (distance) to the reference instances and class imbalance ratio to select the most confident classifier for a given observation are presented. Both approaches come in two modes, first one based on the k-Nearest Oracles (KNORA) and the second one also considering classifier mistakes.

3.1 *Diversity Ensemble Pruning*

This section proposes the *Diversity Ensemble Pruning* (DEP) algorithm. Clustering-based ensemble pruning methods, despite possessing a separate taxonomy, are strongly related to the notion of static classifier selection. The main novelty of the presented approach is the clustering criterion based on the influence of individual base classifiers on the entire ensemble diversity. Thanks to this, it is possible to group the base models in a one-dimensional diversity space. This algorithm, originally proposed by the author of the following thesis to deal with balanced problems [283], has been experimentally evaluated for the imbalanced data classification task. The goal here is to test whether classifier selection methods, which employ diversity measures in order to find the most competent models in a given region of the feature space, can improve the ensemble's ability to detect minority class instances without the use of data preprocessing techniques.

Clustering criterion

Here, the measure used for creating the space for the clustering-based pruning is proposed. As the non-pairwise and averaged pairwise diversity measures consider all the base models together and calculate one value for the entire ensemble, thus they could not be used for pruning, because they do not present an impact of a particular base classifier on the ensemble diversity. Therefore a novel measure H as the clustering criterion is proposed, which is the difference between the value of diversity measure for the whole ensemble Π and the value of diversity for the ensemble without a given classifier Ψ_i [283].

$$H(\Psi_i) = Div(\Pi) - Div(\Pi - \Psi_i). \tag{3.1}$$

Thanks to this proposition the impact of each base learner on the ensemble diversity is presented in a one-dimensional space, shown in Fig. 3.1.

Diversity based one-dimensional clustering space and cluster pruning

The chosen clustering algorithm is applied to the obtained clustering space. The pruned ensemble consists of the base models with the highest *balanced accuracy score* selected from each cluster. Then, the final decision is made based on *support accumulation* of selected prototype classifiers using the sum rule [76] shown in Equation 2.31 on p. 32.

The *k-means* clustering algorithm [162, 168] has been employed to find a set number of clusters in the clustering space constructed by the proposed H measure. From each group a representative classifier with the highest predictive performance has been chosen. The goal is to construct an ensemble containing strong, yet diverse base models, as these two characteristics are distinguishing features of a well-performing classifier ensemble. Pseudocode for the proposed method is presented in Algorithm 1.

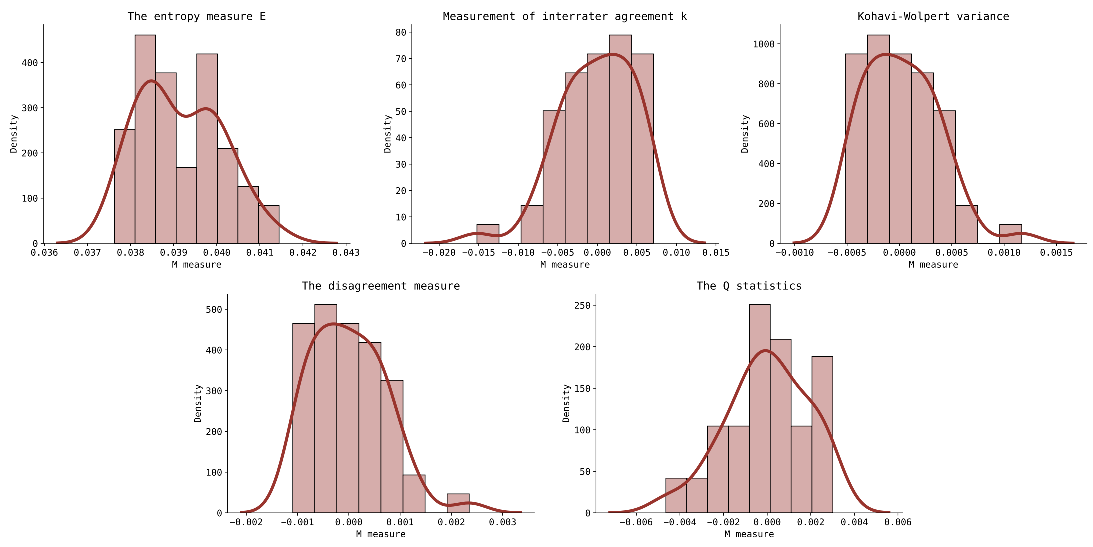


Figure 3.1: *Histograms and density estimation plots for H measure based on each ensemble diversity metric calculated on the glass2 dataset.*

Algorithm 1 Pseudocode of the proposed DEP algorithm

```
Input:
    $\Pi = \{\Psi_1, \Psi_2, \ldots, \Psi_n\}$ – classifier pool,
    $c$ – number of clusters,
    $\mathcal{LS}$ – learning set,
Symbols:
    $\mathcal{H}$ – set of $H$ measure values for each base classifier,
    $\mathcal{C}$ – set of clusters,
    $\mathcal{S}$ – set of evaluation metric values for each base classifier,
Output:
    $\Pi_S$ – pool of selected classifiers.

1: $\mathcal{H} \leftarrow \varnothing, \Pi_S \leftarrow \varnothing, \mathcal{S} \leftarrow \varnothing$
2: for each $\Psi_i$ in $\Pi$ do
3:     $\mathcal{H} \leftarrow H_i = \text{DIV}(\Pi, \mathcal{LS}) - \text{DIV}((\Pi - \Psi_i), \mathcal{LS})$
4:     $\mathcal{S} \leftarrow \text{BAC}_i = \text{EVALUATE}(\Psi_i, \mathcal{LS})$
5: end for
6: $\mathcal{C} = \text{K-MEANS}(\mathcal{H}, c)$
7: for each cluster $C_j$ in $\mathcal{C}$ do
8:     $\Pi_S \leftarrow \text{SELECT}(\Pi, C_j, \mathcal{S})$
9: end for
```

The description of the functions used in the pseudocode is as follows:

- DIV() – calculates the ensemble diversity of a given classifier pool Π based on the provided learning set $\mathcal{LS}$.
- EVALUATE() – calculates the *balance accuracy score* on learning set $\mathcal{LS}$ for each base classifier Ψ_i in order to use it later in the selection process.

- k-Means() – carries out the clustering process of a given one-dimensional diversity space $\mathcal{H}$ into c clusters using the *k-means* algorithm. Returns information about the cluster each base classifier belongs to.
- select() – from each cluster C_j selects a prototype classifier with the highest *balanced accuracy score* to be a part of the new classifier pool Π_S.

Computational and memory complexity analysis

The proposed method includes the stage of determining the H measure value of each base classifier, the clustering of models in the diversity space and the selection of prototype classifiers.

In order to obtain the H measure value for each base classifier, first, the ensemble diversity must be calculated. The complexity of this process is $O(n)$ or $O(n^2)$, where n is the number of base classifiers, depending on whether the non-pairwise or pairwise measure is used. Then, the H measure calculation process has the complexity of $O(n)$.

The *k-means* algorithm was used for clustering in diversity space. Therefore, the complexity of clustering is $O(ncde)$, where c is the number of clusters, d is the number of data dimensions, and e describes the number of iterations/epochs of the algorithm [26]. As the clustering space is one-dimensional, complexity is reduced to $O(nce)$.

3.1.1 Experimental evaluation

This subsection presents the motivation, goals and set-up of the performed experiments, as well as their results.

Research questions

The conducted research aims to answer two main questions:

Q1. Is the static classifier selection able to improve the results obtained by combining the entire classifier pool for the task of imbalanced data classification?

Q2. Can the use of static classifier selection in the problem of imbalanced data classification result in performance comparable with the use of preprocessing techniques?

Goals of the experiments

Experiment 1 – Parametrization

The aim of the first experiment is to determine the number of clusters for which the methods based on each of the measures of diversity and the base classifier performs best. Parameterization is carried out on the basis of the *balanced accuracy score*, and the best pairs of the diversity measure and the number of clusters are used in the next experiments.

Experiment 2 – Comparison with standard combination
The aim of the second experiment is to compare the previously selected methods with a combination of the entire classifier pool. *Support accumulation* and *majority voting* of all 50 base models were used as reference methods. The best of the proposed methods is then used in Experiment 3.

Experiment 3 – Comparison with preprocessing techniques
In the third experiment, the method selected in Experiment 2 is compared with the combination of the whole classifier pool generated using preprocessing methods. Preprocessing is performed separately for each of the bootstraps generated by *Stratified Bagging.*

Experimental set-up
The research was carried out on 41 imbalanced datasets presented in Table 2.3 on p. 44. However, it should be noted that the experiments could only be carried out on those datasets for which the *k-means* clustering algorithm was able to find the desired number of clusters (from 2 to 7) for a set classification algorithm and diversity measure.

The evaluation of the proposed methods is based on six metrics widely used in the case of imbalanced classification problems. Three popular classification algorithms were used as base models, ensemble diversity was calculated using five different measures, and four preprocessing techniques were used as reference methods. Detailed information is presented below:

- Evaluation measures – *balanced accuracy score* (BAC), $Gmean_s$, F_1 *score*, *precision*, *recall*, and *specificity*,
- Classification algorithms – *Gaussian Naïve Bayes* classifier (GNB), *k-Nearest Neighbors* classifier (*k*NN), and *Classification and Regression Tree* (CART),
- Ensemble diversity measures – *The entropy measure* E, *Kohavi-Wolpert variance* (KW), *measurement of interrater agreement* k, *the averaged* Q *statistics* (Q_{av}), and *the averaged disagreement measure* (Dis_{av}),

- Reference methods:
  - *Stratified Bagging* without preprocessing – *Majority Voting* (MV), *Support Accumulation* (SACC),
  - *Stratified Bagging* paired with preprocessing (SACC only) – *Random Oversampling* (ROS), SMOTE, SVM-SMOTE (SVM) and *Borderline*-SMOTE (B2).

The fixed size of the classifier pool was set to 50 base models, generated using a stratified version of *Bagging* [30]. This *Bagging* generates each bootstrap sampling with replacement majority and minority classes separately while maintaining the original imbalance ratio. The size of each bootstrap is set to half the size of the original training set. The proposed approaches were evaluated on the basis of 5 times repeated 2-fold cross-validation. The ensemble's decision is based on *support accumulation*. Statistical analysis of the obtained results was performed using the Wilcoxon global rank test [62]. All experiments have been implemented in *Python* programming language and can be repeated using the code on *Github*[1].

Experiment 1 – Parametrization

Table 3.1 presents the results of the cluster number parametrization for each classifier diversity measure in relation to the type of base classifier. The digit after CL denotes the set number of clusters. The numbers under the average rank of each method indicate, which algorithms were statistically significantly worse than the one in question.

In the case of GNB, there is a clear tendency for methods using 2 or 3 clusters to achieve the best results, regardless of the diversity measure used. The kNN classifier performs best when *k-means* divides clustering space into two groups. A more interesting situation can be observed in the case of the CART classifier, which performs best in the case of an odd number of clusters, with an emphasis on 3 and 5 groups.

Based on the results obtained and the statistical tests conducted, the following pairs of the measure of diversity and number of clusters were selected for the next experiment:

- GNB – E: 2, k: 2, KW: 2, Dis_{av}: 2, Q_{av}: 3,
- kNN – E: 2, k: 2, KW: 2, Dis_{av}: 2, Q_{av}: 4,
- CART – E: 5, k: 3, KW: 3, Dis_{av}: 3, Q_{av}: 5.

Experiment 2 – Comparison with standard combination

Figure 3.2 shows radar plots with the average ranks achieved by each method on all evaluation metrics.

[1] `https://github.com/w4k2/iccs21-ensemble-pruning`

Table 3.1: *Results of Wilcoxon statistical test on global ranks for each measure of diversity and number of clusters. Calculated based on* BAC. *The higher the average rank value, the better.*

GNB						
	DEP-CL2	DEP-CL3	DEP-CL4	DEP-CL5	DEP-CL6	DEP-CL7
	(1)	(2)	(3)	(4)	(5)	(6)
E	4.696	4.089	3.804	2.679	3.179	2.554
	3, 4, 5, 6	4, 5, 6	4, 6	—	—	—
k	4.679	4.018	3.446	3.036	3.000	2.821
	3, 4, 5, 6	4, 5, 6	—	—	—	—
KW	4.679	4.018	3.464	3.054	2.946	2.839
	3, 4, 5, 6	4, 5, 6	—	—	—	—
Dis_{av}	4.679	4.018	3.464	3.054	2.946	2.839
	3, 4, 5, 6	4, 5, 6	—	—	—	—
Q_{av}	4.089	4.339	3.286	3.375	3.125	2.786
	5, 6	3, 4, 5, 6	—	—	—	—
kNN						
	DEP-CL2	DEP-CL3	DEP-CL4	DEP-CL5	DEP-CL6	DEP-CL7
	(1)	(2)	(3)	(4)	(5)	(6)
E	4.054	3.893	3.286	3.482	3.250	3.036
	6	6	—	—	—	—
k	4.268	3.607	2.857	3.679	3.000	3.589
	3, 5	—	—	—	—	—
KW	4.268	3.607	2.857	3.679	3.000	3.589
	3, 5	—	—	—	—	—
Dis_{av}	4.268	3.607	2.857	3.679	3.000	3.589
	3, 5	—	—	—	—	—
Q_{av}	3.339	3.393	3.929	3.839	3.179	3.321
	—	—	—	—	—	—
CART						
	DEP-CL2	DEP-CL3	DEP-CL4	DEP-CL5	DEP-CL6	DEP-CL7
	(1)	(2)	(3)	(4)	(5)	(6)
E	2.103	4.241	3.172	4.310	3.034	4.138
	—	1, 3, 5	1	1, 3, 5	1	1, 3, 5
k	2.276	4.138	3.138	3.983	3.638	3.828
	—	1, 3	1	1, 3	1	1
KW	2.276	4.155	3.172	4.017	3.672	3.707
	—	1, 3	1	1, 3	1	1
Dis_{av}	2.276	4.155	3.172	4.052	3.672	3.672
	—	1, 3	1	1, 3	1	1
Q_{av}	1.948	4.448	3.069	4.672	3.328	3.534
	—	1, 3, 5, 6	1	1, 3, 5, 6	1	1

For the *gaussian naïve bayes* classifier, the advantage of the proposed methods over the combination of the entire available classifier pool can be observed. The only exception is *recall*, where DEP-E2 is comparable to the reference methods, while the other proposed approaches display a slightly lower average rank value.

These observations are confirmed by Table 3.2. It presents the results of the performed statistical analysis, on the basis of which it can be concluded that the proposed methods achieve statistically significantly better average ranks than the combination of the entire classifier pool for each of the metrics, except *recall*, where no statistically significant differences were reported. Worth noting is also the identical performance of methods based on measures k, KW, and Dis_{av}.

In the case of the kNN classifier, the achieved results again speak in favor of the proposed methods. *Precision* achieved by *support accumulation* of the entire pool of classifiers is comparable to that achieved by the ensemble pruning algorithms. However, the advantage obtained in terms of *recall* while maintaining similar *precision* proves that the

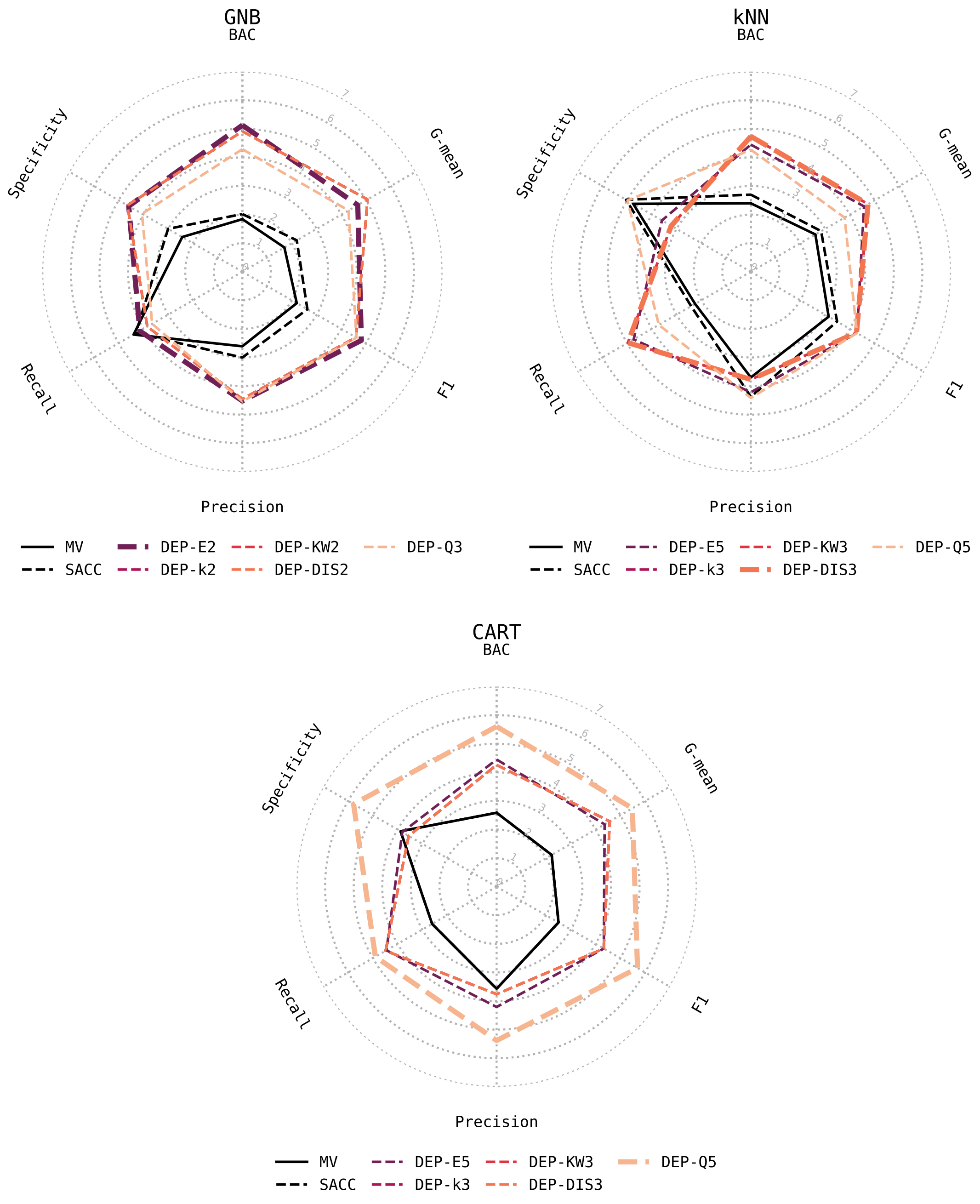


Figure 3.2: *Visualization of the mean ranks achieved by each method.*

proposed methods are oriented towards recognizing the minority class. This is especially visible in the case of measures k, KW and Dis_{av}, which again show exactly the same performance.

The results of the statistical analysis for kNN classifier are also slightly more interesting. There is a statistically significant advantage of the proposed solutions over the combination of the entire pool in the case of BAC, *Gmean*$_s$ and *recall* (at the expense of *specificity*). When it comes to F_1 *score*, the ensemble pruning algorithms are statistically

Table 3.2: *Results of Wilcoxon statistical test on global ranks for proposed methods in comparison to the combination of the whole classifier pool. The higher the average rank value, the better.*

				GNB			
	MV (1)	SACC (2)	DEP-E2 (3)	DEP-k2 (4)	DEP-KW2 (5)	DEP-DIS2 (6)	DEP-Q3 (7)
BAC	1.839	2.018	5.125	4.911	4.911	4.911	4.286
	—	—	1, 2	1, 2	1, 2	1, 2	1, 2
Gmean$_s$	1.696	2.196	4.661	5.054	5.054	5.054	4.286
	—	1	1, 2	1, 2	1, 2	1, 2	1, 2
F_1 score	2.196	2.625	4.804	4.589	4.589	4.589	4.607
	—	—	1, 2	1, 2	1, 2	1, 2	1, 2
precision	2.607	3.000	4.518	4.446	4.446	4.446	4.536
	—	—	1, 2	1, 2	1, 2	1, 2	1, 2
recall	4.393	4.304	4.143	3.839	3.839	3.839	3.643
	—	—	—	—	—	—	—
specificity	2.429	3.000	4.589	4.643	4.643	4.643	4.054
	—	1	1, 2	1, 2	1, 2	1, 2	1, 2
				*k*NN			
	MV (1)	SACC (2)	DEP-E2 (3)	DEP-k2 (4)	DEP-KW2 (5)	DEP-DIS2 (6)	DEP-Q4 (7)
BAC	2.393	2.696	4.446	4.732	4.732	4.732	4.268
	—	—	1, 2	1, 2	1, 2	1, 2	1, 2
Gmean$_s$	2.607	2.839	4.571	4.732	4.732	4.732	3.786
	—	—	1, 2	1, 2	1, 2	1, 2	1, 2
F_1 score	3.143	3.482	4.286	4.268	4.268	4.268	4.286
	—	—	1	1	1	1	1
precision	3.696	4.375	4.214	3.768	3.768	3.768	4.411
	—	—	—	—	—	—	—
recall	2.250	2.411	4.732	4.964	4.964	4.964	3.714
	—	—	1, 2, 7	1, 2, 7	1, 2, 7	1, 2, 7	1, 2
specificity	4.750	5.036	3.589	3.214	3.214	3.214	4.982
	4, 5, 6	3, 4, 5, 6	—	—	—	—	3, 4, 5, 6
				CART			
	MV (1)	SACC (2)	DEP-E5 (3)	DEP-k3 (4)	DEP-KW3 (5)	DEP-DIS3 (6)	DEP-Q5 (7)
BAC	2.586	2.586	4.448	4.259	4.259	4.259	5.603
	—	—	1, 2	1, 2	1, 2	1, 2	all
Gmean$_s$	2.224	2.224	4.362	4.569	4.569	4.569	5.483
	—	—	1, 2	1, 2	1, 2	1, 2	all
F_1 score	2.500	2.500	4.328	4.328	4.328	4.328	5.690
	—	—	1, 2	1, 2	1, 2	1, 2	all
precision	3.569	3.569	4.207	3.759	3.759	3.759	5.379
	—	—	—	—	—	—	all
recall	2.603	2.603	4.448	4.483	4.483	4.483	4.897
	—	—	1, 2	1, 2	1, 2	1, 2	1, 2
specificity	3.879	3.879	3.810	3.552	3.552	3.552	5.776
	—	—	—	—	—	—	all

significantly better than majority voting, but the result obtained by them is comparable to *support accumulation*.

Particularly promising results can be observed when using CART as the base classifier. In this case, the measure of diversity Q_{av} performs best. Based on the statistical analysis presented in Table 3.2, it achieves statistically significantly better results than the combination of the entire classifier pool, as well as the pruning algorithms using other measures of diversity for the clustering space construction. This is true for every metric except *recall*.

Based on the results of the statistical analysis, the GNB DEP-E2, *k*NN DEP-DIS2, and CART DEP-Q5 methods were selected for the next experiment. These approaches displayed the highest average ranks as well as a good ability to recognize the minority class.

Experiment 3 – Comparison with preprocessing techniques

Figure 3.3 shows the results of comparing the methods selected in Experiment 2 with the approaches employing preprocessing techniques.

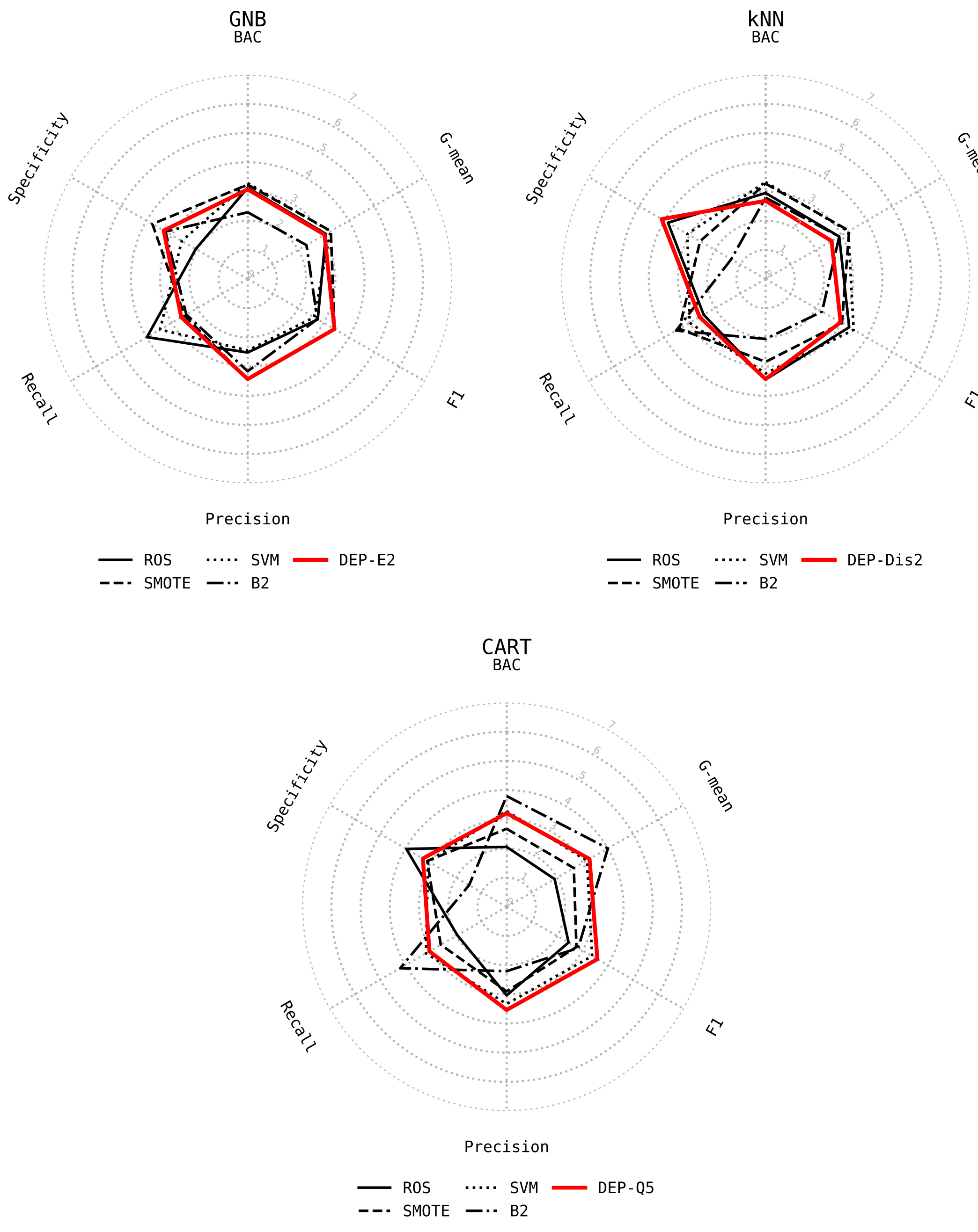


Figure 3.3: *Visualization of the mean ranks achieved by each method.*

When the base classifiers are GNB and kNN, it can be noticed that, despite achieving average rank values for each of the metrics, the proposed methods are never statistically significantly worse than the reference approaches using preprocessing (Table 3.3). Additionally, GNB DEP-E2 shows statistically higher *precision* than that achieved by using

Table 3.3: *Results of Wilcoxon statistical test on global ranks for the selected methods in comparison to the preprocessing techniques. The higher the average rank value, the better.*

			GNB		
	ROS	SMOTE	SVM	B2	DEP-E2
	(1)	(2)	(3)	(4)	(5)
BAC	3.125	3.232	3.286	2.286	3.071
	4	4	4	—	—
Gmean$_s$	3.089	3.286	3.268	2.321	3.036
	4	4	4	—	—
F_1 score	2.768	3.429	2.625	2.750	3.429
	—	3	—	—	—
precision	2.518	3.446	2.446	3.161	3.429
	—	1, 3	—	—	1, 3
recall	3.982	2.500	3.464	2.429	2.625
	2, 4, 5	—	2, 4	—	—
specificity	2.054	3.768	2.607	3.250	3.321
	—	1, 3	—	1	1
			*k*NN		
	ROS	SMOTE	SVM	B2	DEP-DIS2
	(1)	(2)	(3)	(4)	(5)
BAC	2.946	3.268	3.321	2.786	2.679
	—	—	—	—	—
Gmean$_s$	2.911	3.304	3.268	2.911	2.607
	—	—	—	—	—
F_1 score	3.304	3.018	3.482	2.232	2.964
	4	4	4	—	—
precision	3.446	2.839	3.232	2.054	3.429
	2, 4	4	4	—	4
recall	2.446	3.411	3.000	3.536	2.607
	—	1	1	1	—
specificity	3.857	2.589	3.089	1.357	4.107
	2, 3, 4	4	2, 4	—	all
			CART		
	ROS	SMOTE	SVM	B2	DEP-Q5
	(1)	(2)	(3)	(4)	(5)
BAC	2.052	2.672	3.276	3.793	3.207
	—	—	1, 2	1, 2	1
Gmean$_s$	1.897	2.655	3.172	4.000	3.276
	—	1	1	1, 2, 3	1
F_1 score	2.448	2.759	3.379	2.828	3.586
	—	—	1, 2	—	1, 4
precision	3.034	2.897	3.328	2.207	3.534
	4	4	4	—	4
recall	1.948	2.603	3.190	4.207	3.052
	—	1	1, 2	all	—
specificity	3.966	3.138	3.103	1.483	3.310
	2, 3, 4	4	4	—	4

Random Oversampling and SVM-SMOTE, and *k*NN DEP-DIS2 achieves better precision than the ensemble using *Borderline*-SMOTE for data preprocessing.

The ensemble pruning methods seem to perform better when using the CART decision tree as the base classifier. Again, none of the reference methods achieved statistically significantly better average ranks than the proposed approach. At the same time, however, CART DEP-Q5 achieves a statistically significantly better rank value than ROS for BAC, *Gmean$_s$* and *F_1 score*. This method is also statistically significantly better than *Borderline*-SMOTE in terms of *F_1 score* and *specificity*.

Observations

Based on the results of Experiment 1, it can be concluded that the classifier pool generation using *Stratified Bagging* probably does not allow for achieving a high ensemble diversity in the case of GNB and kNN base classifiers. This is indicated by the fact that the methods using these classifiers perform best when the clustering space is divided into just two groups. Decision trees, which show a greater tendency to obtain diverse base models, do much better in this respect. It is also worth noting that in the case of CART, due to no tree depth limitation, the results of the majority vote were in line with the accumulation of support.

Regardless of the base classifier used, the results obtained with the use of the measures of diversity k, KW, and Dis_{av} were exactly the same. On this basis, it can be concluded that the diversity spaces generated on their basis coincide. An example of this can be seen in the example shown in Figure 3.1, where all three spaces have the same distribution density (where the space based on *measurement of interrater agreement* k is a mirror image of the spaces based on KW and Dis_{av}).

Experiment 2 proved that by a skillful selection of a small group of classifiers, in the imbalanced data classification problem, it is possible to achieve a better performance than that achieved by combining the decisions of the entire classifier pool.

Experiment 3 was able to confirm that thanks to employing the classifier selection methods to the problem of imbalanced data classification, it is possible to obtain results statistically not worse (and sometimes statistically significantly better) than those achieved by the ensembles using preprocessing techniques.

Although, in the case of decision trees, conducted statistical tests indicate that the most suitable diversity measure for the problems considered during experimentation may be the averaged Q statistics, it can not definitively be considered the best. As stated in [151], after studying various diversity measures, there is no definitive connection between the measures and the performance improvement. Nonetheless Q_{av} was recommended only based on ease of interpretation and calculation.

Answers to research questions

The answers to the previously formulated research questions are as follows:

Q1. Is the static classifier selection able to improve the results obtained by combining the entire classifier pool for the task of imbalanced data classification?

A1. The conducted experiments have shown that the use of a static classifier selection, based on ensemble diversity, is able to statistically significantly improve the ensemble performance in the task of the imbalanced data classification.

Q2. Can the use of static classifier selection in the problem of imbalanced data classification result in performance comparable with the use of preprocessing techniques?

A2. The obtained results confirmed that classifier selection algorithms may show statistical dependency to the approaches using preprocessing techniques in the task of imbalanced data classification.

3.2 Clustering-based multistage organization

The following section introduces two proposals for the extension of the *multistage majority voting organization* (MOMV) originally proposed by Ruta and Gabryś [201] and described in more detail in Chapter 2. Both approaches are strongly based on the classifier clustering in one-dimensional diversity space, which was introduced in Section 3.1 and follow the same procedure of calculating the H measure (Equation 3.1). Although the *multistage organization* is not the main subject of the thesis, it was considered an interesting complement to the proposed DEP algorithm.

Two-step majority voting organization (TSMV)

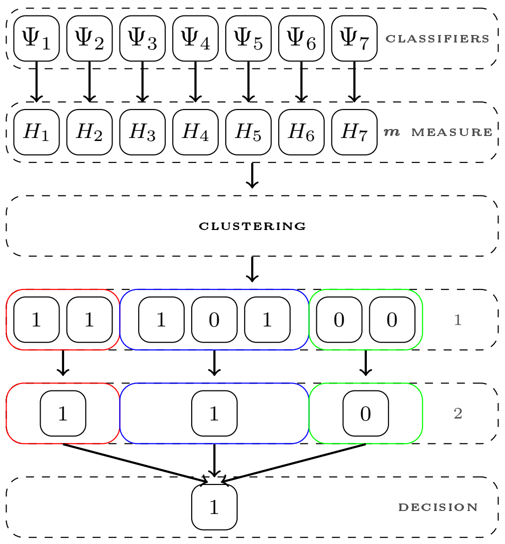
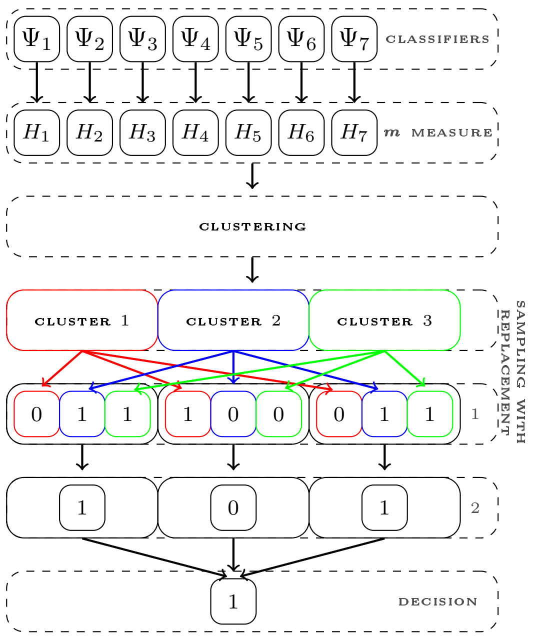

CLASSIFIERS
Ψ_1 Ψ_2 Ψ_3 Ψ_4 Ψ_5 Ψ_6 Ψ_7
H_1 H_2 H_3 H_4 H_5 H_6 H_7
m MEASURE
CLUSTERING
1 1 1 0 1 0 0
1
1 1 0
2
1
DECISION

Figure 3.4: *Example of a two-step majority voting organization with 9 classifiers divided into 3 clusters.* Layer 2 *is the result of majority voting of each cluster and the final decision is made by the second majority voting.*

The first proposed method, called *Two-step Majority Voting Organization* (TSMV), is a modification of the MOMV structure described in [201]. Instead of allocating outputs to different groups by permutation, the base models in each cluster are treated as a separate ensemble combined by *majority voting*. The calculation of H measure as well as clasterizaton process are conducted in the same fashion as in the PDE. As the remainder, the procedure is described in Algorithm 2. Then, predictions from all clusters are collected and the *majority voting* rule is applied for the second time, in order to obtain a final decision. This process is depicted in Figure 3.4 and the pseudocode for the prediction process of TSMV is presented in Algorithm 3.



Ψ_1 Ψ_2 Ψ_3 Ψ_4 Ψ_5 Ψ_6 Ψ_7
CLASSIFIERS
H_1 H_2 H_3 H_4 H_5 H_6 H_7
m MEASURE
CLUSTERING
CLUSTER 1
CLUSTER 2
CLUSTER 3
SAMPLING WITH REPLACEMENT
0 1 1
1 0 0
0 1 1
1
1
0
1
2
1
DECISION

Figure 3.5: *Example of two-step majority voting organization with 9 classifiers divided into 3 clusters, using sampling with replacement. The number of groups and classifiers in each group in the first layer is equal to the number of clusters found.* Layer 2 *and the final decision are also made according to the majority voting.*

Algorithm 2 Pseudocode of the clustering process for TSMV and RSMO methods

```
Input:
    $\Pi = \{\Psi_1, \Psi_2, \ldots, \Psi_n\}$ – classifier pool,
    $c$ – number of clusters,
    $\mathcal{LS}$ – training set,
Symbols:
    $\mathcal{H}$ – set of $H$ measure values for each base classifier,
Output:
    $\mathcal{C} = \{C_1, C_2, \ldots, C_c\}$ – set of clusters.

1: $\mathcal{H} \leftarrow \varnothing$
2: for each $\Psi_i$ in $\Pi$ do
3:     $\mathcal{H} \leftarrow H_i = \text{DIV}(\Pi, \mathcal{LS}) - \text{DIV}((\Pi - \Psi_i), \mathcal{LS})$
4: end for
5: $\mathcal{C} = \text{K-MEANS}(\mathcal{H}, c)$
```

The second proposed method, called *Random Sampling Multistage Organization* (RSMO), is a modification pf TSMV introducing *sampling with replacement* before the first voting step. This approach is based on the assumption that classifiers belonging to the same cluster make similar decisions, so they don't have to be all used during the the classification process. In RSMO, the first layer of voting is constructed by generating a number

Algorithm 3 Prediction pseudocode of the TSMV

```
Input:
    $\Pi = \{\Psi_1, \Psi_2, \ldots, \Psi_n\}$ – classifier pool,
    $\mathcal{C} = \{C_1, C_2, \ldots, C_c\}$ – set of clusters.
    $\mathcal{TS}$ – testing set,
Symbols:
    $\mathcal{V}$ – set of majority voting results.
Output:
    $Decision$ – classification results.

1: for each cluster $C_j$ in $\mathcal{C}$ do
2:     $\Pi_{Cj} = \{\forall i \in C_j, \Psi_i\}$
3:     $\mathcal{V} \leftarrow$ MAJORITYVOTING($\Pi_{Cj}, \mathcal{TS}$)        ▷ First voting
4: end for
5: $Decision$ = MODE($\mathcal{V}$)        ▷ Second voting
```

of groups equal to the number of clusters c. Each group contains one classifier sampled from each of the clusters found. Example of *random sampling multistage organization* is shown in Figure 3.5 and the pseudocode for its prediction process is presented in Algorithm 3.

Algorithm 4 Prediction pseudocode of the RSMO

```
Input:
    $\Pi = \{\Psi_1, \Psi_2, \ldots, \Psi_n\}$ – classifier pool,
    $\mathcal{C} = \{C_1, C_2, \ldots, C_c\}$ – set of clusters.
    $\mathcal{TS}$ – testing set,
    $c$ – number of clusters,
Symbols:
    $\mathcal{V}$ – set of majority voting results.
Output:
    $Decision$ – classification results.

1: for each cluster $C_j$ in $\mathcal{C}$ do
2:     for $k$ in range($c$) do
3:         $\Pi_{C_j} \leftarrow$ SAMPLING($\Pi, C_k$)        ▷ Sampling with replacement
4:     end for
5:     $\mathcal{V} \leftarrow$ MAJORITYVOTING($\Pi_{C_j}, \mathcal{TS}$)        ▷ First voting
6: end for
7: $Decision$ = MODE($\mathcal{V}$)        ▷ Second voting
```

The following functions are used in the presented pseudocodes:

- DIV() – calculates the ensemble diversity of a given classifier pool Π based on the provided learning set $\mathcal{LS}$.

- K-MEANS() – carries out the clustering process of a given one-dimensional diversity space $\mathcal{H}$ into c clusters using the *k-means* algorithm. Returns information about the cluster each base classifier belongs to.
- MAJORITYVOTING() – uses all classifiers belonging to a given pool Π_{Cj} to classify the instances in the testing set $\mathcal{TS}$ using *majority voting.*
- MODE() – returns the modal (most common) value in a set $\mathcal{V}$.
- SAMPLING() – select, using *sampling with replacement*, a single classifier Ψ_i from a given cluster.

Computational and memory complexity analysis

Similar to the PDE methods proposed in Section 3.1, TSMV and RSMO include the stage of the H measure calculation as well as clustering of base models in the prepared space.

The computational complexity of diversity calculation is again $O(n)$ or $O(n^2)$, where n is the number of base classifiers, depending on whether the non-pairwise or pairwise measure is used. The complexity of H measure calculation process is $O(n)$.

The complexity of the *k-means* clustering algorithm is $O(ncde)$, where c is the number of clusters, d is the number of data dimensions, and e describes the number of iterations/epochs [26]. Complexity is reduced to $O(nce)$, as the clustering space is one-dimensional.

During the prediction step, TSMV first performs *majority voting* for each classifier pool Π_{C_i} with complexity $O(n_{C_i}+ \mid M \mid)$, where n_{C_i} denotes number of base models in pool Π_{C_i} and $\mid M \mid$ denotes the number of classes. In the case of binary classification $\mid M \mid$ can be omitted, resulting in $O(n_{C_i})$ complexity. Then, the *mode* operation with complexity $O(c)$ is used to obtain the final decision.

For RSMO, *sampling with replacement* is performed for each cluster C_i with complexity $O(\mid C_i \mid)$, where $\mid C_i \mid$ is a cardinality of cluster C_i. Then, for each classifier pool Π_{C_i}, *majority voting* is carried out with complexity $O(c+ \mid M \mid)$. Finally, *Mode* operation with complexity $O(c)$ is again employed to obtain prediction.

3.2.1 Experimental evaluation

Here, the motivation, goals and set-up of the performed experiments are presented.

Research questions

The conducted research aims to answer the following main questions:

Q1. Can the use of a clustering-based two-stage majority voting structure improve the performance of the imbalanced data classification?

Q2. Can the introduction of sampling with replacement before the first voting stage allow to increase the generalization ability of the MOMV structure?

Q3. Can the proposed multistage majority voting organization compete with methods employing preprocessing techniques?

Goals of the experiments

Experiment 1 – Comparison with standard combination

The aim of the first experiment is to check how the proposed two-step majority voting methods compare to a simple, one-step, combination of a classifier pool.

Experiment 2 – Comparison with preprocessing techniques

In the second experiment, the methods selected in Experiment 1 will be compared with preprocessing-based reference methods.

Experimental set-up

The research was carried out on 41 imbalanced datasets presented in Table 2.3 on p. 44. Since the TSMV and RSMO algorithms are based on the same clustering approach as PDE, again the experiments could only be carried out on those datasets for which the *k-means* clustering was able to find the set number of clusters (ranged from 2 to 7) for a set pair of diversity measure and classification algorithm.

Since the evaluated methods are strongly based on the one-dimensional diversity space introduced in Section 3.1, the experimental set-up is almost identical to that described for PDE algorithm. However, taking into account the fact that *multistage majority voting organization* is not the main interest of this thesis, but only an extension of the previously studied method, the experimental evaluation was reduced to two base classifiers. Details on used set-up are listed below:

- Evaluation measures – *balanced accuracy score* (BAC), $Gmean_s$, F_1 *score*, *precision*, *recall*, and *specificity*,
- Classification algorithms – *Gaussian Naïve Bayes* classifier (GNB) and *Classification and Regression Tree* (CART),
- Ensemble diversity measures – *The entropy measure* E, *Kohavi-Wolpert variance* (KW), *measurement of interrater agreement* k, *the averaged Q statistics* (Q_{av}), and *the averaged disagreement measure* (Dis_{av}),

- Reference methods:
  - *Stratified Bagging* without preprocessing – *Majority Voting* (MV), *Support Accumulation* (SACC),
  - *Stratified Bagging* paired with preprocessing (SACC only) – *Random Oversampling* (ROS), SMOTE, SVM-SMOTE (SVM) and *Borderline*-SMOTE (B2).

The fixed size of the classifier pool was set to 50 base models, generated using a stratified version of *Bagging* [30]. This *Bagging* generates each bootstrap sampling with replacement majority and minority classes separately while maintaining the original imbalance ratio. The size of each bootstrap is set to half the size of the original training set. The proposed approaches were evaluated on the basis of 5 times repeated 2-fold cross-validation. The ensemble's decision is based on *support accumulation*. Statistical analysis of the obtained results was performed using the Wilcoxon global rank test [62]. The cluster numbers for each diversity measure were selected in the preliminary research:

- TSMV GNB – E: 4, k: 4, KW: 4, Dis_{av}: 4, Q_{av}: 6,
- RSMO GNB – E: 6, k: 6, KW: 6, Dis_{av}: 4, Q_{av}: 5,
- TSMV CART – E: 5, k: 5, KW: 5, Dis_{av}: 5, Q_{av}: 5,
- RSMO CART – E: 7, k: 7, KW: 7, Dis_{av}: 7, Q_{av}: 3.

All experiments have been implemented in *Python* programming language and can be repeated using the code on *Github*[2].

Experiment 1 – Comparison with standard combination

Figure 3.6 and Table 3.4 show the results of using *two-step mojority voting organization*, both without (TSMV) and with sampling (RSMO), compared to the reference methods for GNB classifier. In the case of TMSV, the results are statistically significantly better than those of the standard combination for all metrics except *recall*. Unfortunately, the ability of the proposed methods to detect the minority class turned out to be statistically significantly worse than that of the reference methods. Noteworthy are the particularly poor results of the method based on the Q_{av} diversity measure.

The results are different when sampling with replacement is introduced to the two-step majority voting. The most significant change occurs for the approach using the Q_{av} diversity measure, which from the worst has become the most balanced for each of

[2]`https://github.com/w4k2/iccs21-ensemble-pruning`

the evaluation metrics. The most important thing is that it has become statistically comparable with the reference methods in terms of the ability to recognize the minority class.

As in the case of GNB, when the CART decision tree is used as the base classifier, the most interesting relationships are represented by the methods based on the averaged Q statistics. Both Figure 3.7 and Table 3.5 show that even without sampling with replacement, the Q_{av}-based method shows the greatest potential in terms of mean ranks. It is, as the only of the proposed approaches, statistically significantly better in terms of BAC than the reference ensemble methods. Additionally, it is statistically significantly the best when it comes to F_1 *score*, *precision* and *specificity*. At the same time, its average rank values in terms of $Gmean_s$ and *recall* are statistically comparable to all other methods.

However, the introduction of sampling with replacement causes that the RSMO approach using Q_{av} for the clustering space definition to become statistically significantly better than most of the other methods – TSMV, MV and SACC – in terms of $Gmean_s$ and *recall*.

On the basis of the obtained results, the following methods were selected for Experiment 2:

- GNB – TSMV-DIS4 and RSMO-Q5,
- CART – TSMV-Q4 and RSMO-Q3.

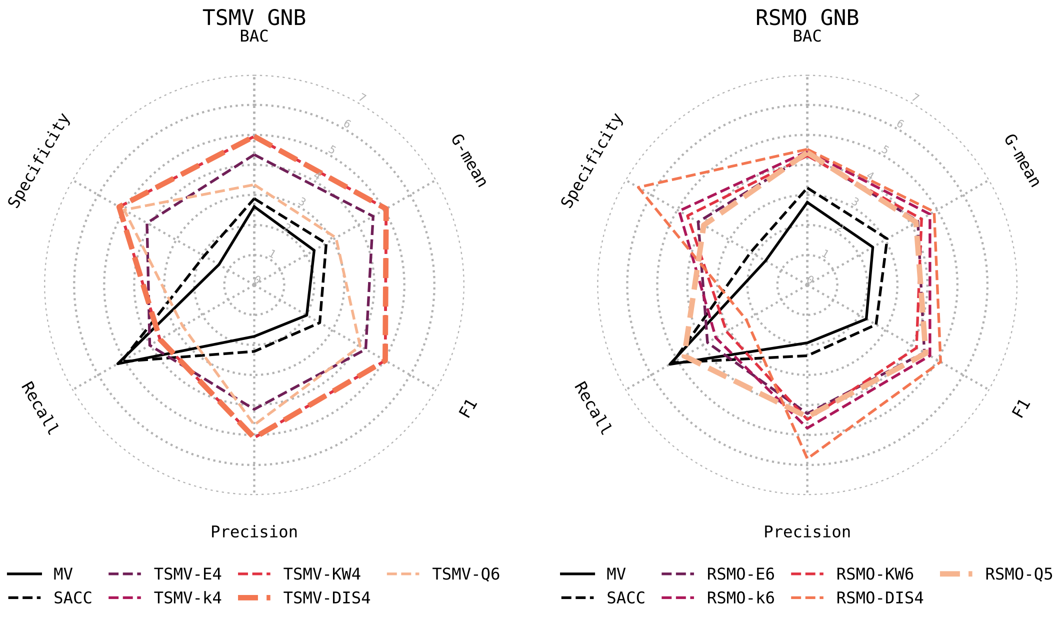


Figure 3.6: *Average rank values for each of the tested methods regarding* GNB.

Table 3.4: *Results of Wilcoxon statistical test on global ranks for proposed methods in comparison to the combination of the whole classifier pool regarding* GNB *classifier. The higher the average rank value, the better.*

			TSMV GNB				
	MV	SACC	E4	k4	KW4	DIS4	Q6
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
BAC	2.607	2.875	4.339	4.946	4.946	4.946	3.339
	–	–	1,2	1,2,7	1,2,7	1,2,7	–
Gmean$_s$	2.304	2.768	4.571	5.071	5.071	5.071	3.143
	–	1	1,2,7	1,2,7	1,2,7	1,2,7	–
F_1 score	2.018	2.518	4.286	5.036	5.036	5.036	4.071
	–	1	1,2	1,2	1,2	1,2	1,2
precision	1.714	2.214	4.143	5.089	5.089	5.089	4.661
	–	1	1,2	1,2,3	1,2,3	1,2,3	1,2
recall	5.250	5.107	4.018	3.625	3.625	3.625	2.750
	3,4,5,6,7	3,4,5,6,7	7	7	7	7	–
specificity	1.357	1.929	4.125	5.196	5.196	5.196	5.000
	–	1	1,2	1,2,3	1,2,3	1,2,3	1,2
			RSMO GNB				
	MV	SACC	E6	k6	KW6	DIS4	Q5
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
BAC	2.750	3.232	4.339	4.446	4.304	4.518	4.411
	—	—	1, 2	1, 2	1, 2	1	1, 2
Gmean$_s$	2.518	3.071	4.250	4.714	4.393	4.875	4.179
	—	—	1, 2	1, 2	1, 2	1, 2	1, 2
F_1 score	2.268	2.643	4.500	4.714	4.196	5.125	4.554
	—	—	1, 2	1, 2	1, 2	1, 2, 5	1, 2
precision	1.929	2.357	4.286	4.768	4.482	5.786	4.393
	—	—	1, 2	1, 2	1, 2	all	1, 2
recall	5.286	5.179	3.839	3.518	3.107	2.339	4.732
	3, 4, 5, 6	3, 4, 5, 6	5, 6	6	6	—	3, 4, 5, 6
specificity	1.607	2.179	4.196	4.929	4.607	6.500	3.982
	—	1	1, 2	1, 2, 3, 7	1, 2	all	1, 2

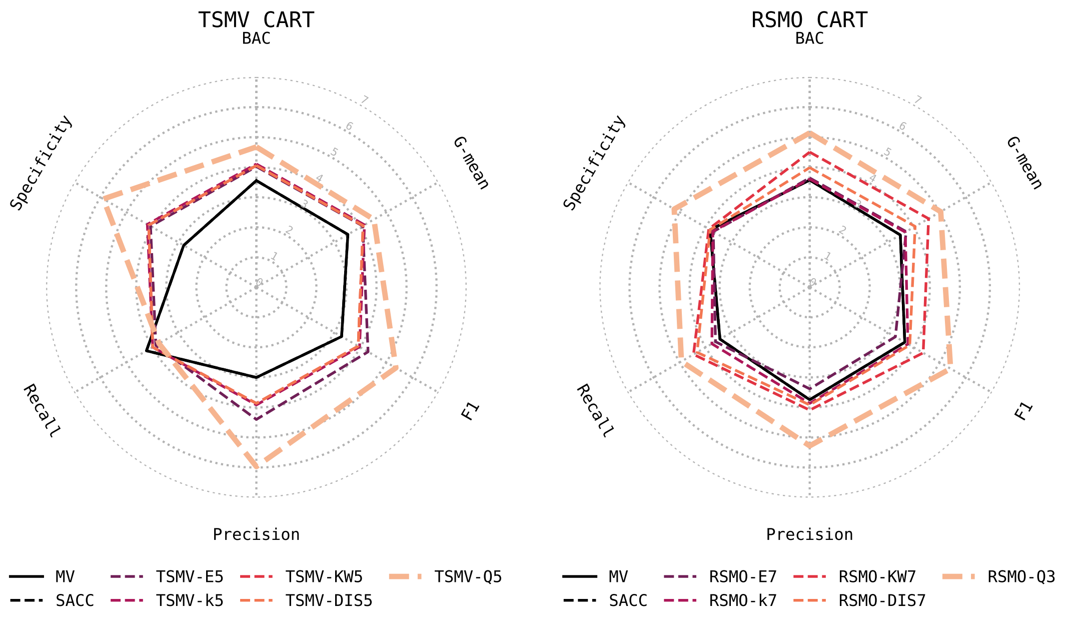


Figure 3.7: *Average rank values for each of the tested methods regarding* CART.

Table 3.5: *Results of Wilcoxon statistical test on global ranks for proposed methods in comparison to the combination of the whole classifier pool regarding* CART *classifier. The higher the average rank value, the better.*

			TSMV CART				
	MV	SACC	E5	k5	KW5	DIS5	Q5
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
BAC	3.552	3.552	4.017	4.103	4.052	4.052	4.672
	—	—	—	—	—	—	1, 2
Gmean$_s$	3.517	3.517	4.086	4.155	4.103	4.103	4.517
	—	—	—	—	—	—	—
F_1 score	3.276	3.276	4.293	3.966	3.914	3.914	5.362
	—	—	—	—	—	—	all
precision	3.000	3.000	4.397	3.914	3.862	3.862	5.966
	—	—	1, 2	1, 2	1, 2	1, 2	all
recall	4.224	4.224	3.862	3.983	3.983	3.983	3.741
	—	—	—	—	—	—	—
specificity	2.793	2.793	4.069	4.190	4.138	4.138	5.879
	—	—	1, 2	1, 2	1, 2	1, 2	all
			RSMO CART				
	MV	SACC	E7	k7	KW7	DIS7	Q3
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
BAC	3.569	3.569	3.638	3.603	4.500	3.983	5.138
	—	—	—	—	1, 2	—	1, 2, 3, 4, 6
Gmean$_s$	3.483	3.483	3.707	3.672	4.569	4.052	5.034
	—	—	—	—	1, 2	—	1, 2, 3, 4, 6
F_1 score	3.655	3.655	3.293	3.776	4.362	3.845	5.414
	—	—	—	—	3	—	all
precision	3.741	3.741	3.379	3.862	4.086	3.914	5.276
	—	—	—	—	—	—	all
recall	3.448	3.448	3.621	3.759	4.466	4.328	4.931
	—	—	—	—	1, 2	—	1, 2, 3, 4
specificity	3.828	3.828	3.707	3.724	3.879	3.828	5.207
	—	—	—	—	—	—	all

Experiment 2 – Comparison with preprocessing techniques

The results of the statistical analysis for the comparison of the TSMV and RSMO with the preprocessing-based approaches are presented in Tables 3.6 and 3.7. Worth noting is the great similarity of both the average rank values and the statistical relationships displayed in comparison with the reference methods by the both algorithms. The average rank values for each of the metrics are shown in Figures 3.8 and 3.9.

When the base classifier is GNB, the proposed methods achieve results comparable to *Borderline*-SMOTE, however, they are statistically significantly worse in terms of *recall* than Random Oversampling and SVM-SMOTE. When the CART decision tree is used as base model for TSMV and RSMO, the the achieved results are statistically significantly better in terms of *precision* than the reference methods. However, the proposed methods are statistically significantly inferior to *Bordeline*-SMOTE in terms of both *Gmean$_s$* and *recall*.

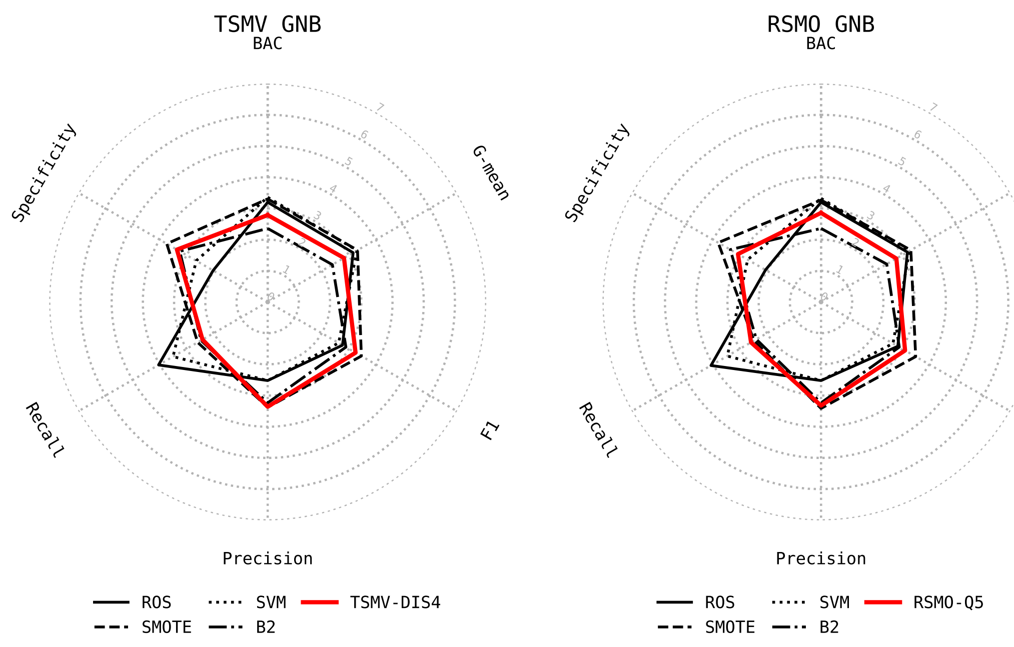


Figure 3.8: *Average rank values for each of the tested methods regarding* GNB.

Table 3.6: *Results of Wilcoxon statistical test on global ranks for the selected methods in comparison to the preprocessing techniques. The higher the average rank value, the better.*

			GNB		
	ROS (1)	SMOTE (2)	SVM (3)	B2 (4)	TSMV-DIS4 (5)
BAC	3.196	3.304	3.357	2.357	2.786
	4	4	4	–	–
Gmean$_s$	3.161	3.321	3.304	2.393	2.821
	4	4	4	–	–
F_1 score	2.768	3.464	2.625	2.893	3.250
	–	3	–	–	–
precision	2.518	3.375	2.518	3.232	3.357
	–	1,3	–	–	–
recall	4.036	2.589	3.500	2.464	2.411
	2,4,5	–	2,4,5	–	–
specificity	2.018	3.732	2.643	3.250	3.357
	–	1,3	–	1	1
	ROS (1)	SMOTE (2)	SVM (3)	B2 (4)	RSMO-Q5 (5)
BAC	3.196	3.268	3.321	2.357	2.857
	4	4	4	—	—
Gmean$_s$	3.196	3.321	3.268	2.429	2.786
	4	4	4	—	—
F_1 score	2.839	3.500	2.661	2.893	3.107
	—	3	—	—	—
precision	2.518	3.411	2.518	3.232	3.321
	—	1, 3	—	—	—
recall	4.071	2.482	3.464	2.393	2.589
	all	—	2, 4, 5	—	—
specificity	2.054	3.804	2.714	3.357	3.071
	—	1, 3	—	1	—

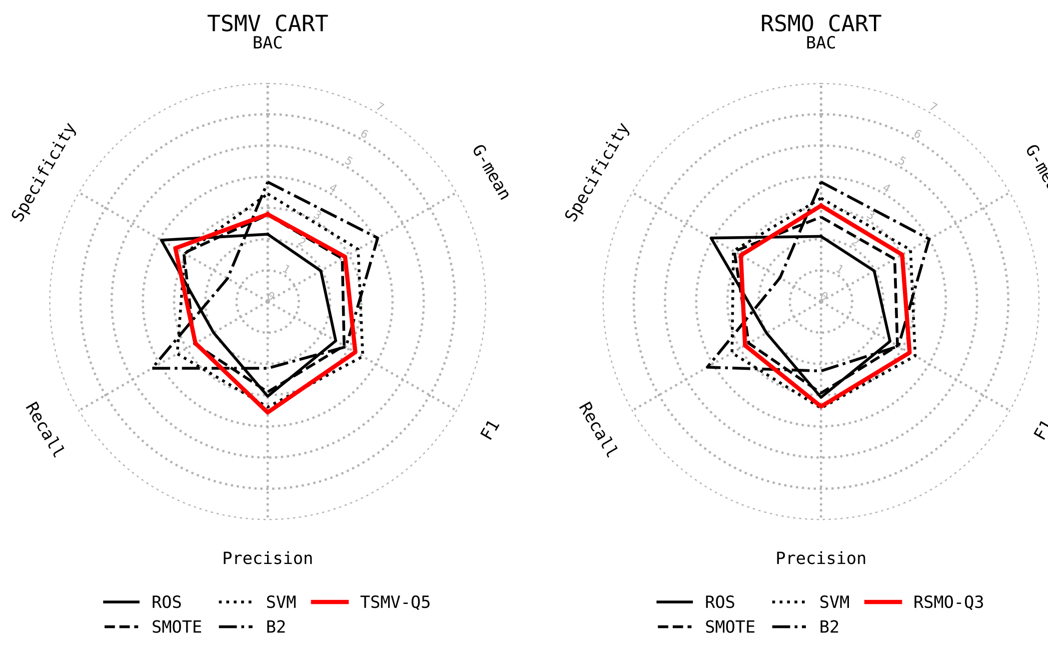


Figure 3.9: *Average rank values for each of the tested methods regarding* CART.

Table 3.7: *Results of Wilcoxon statistical test on global ranks for the selected methods in comparison to the preprocessing techniques. The higher the average rank value, the better.*

	CART				
	ROS	SMOTE	SVM	B2	TSMV-Q5
	(1)	(2)	(3)	(4)	(5)
BAC	2.155	2.776	3.448	3.828	2.793
	—	1	1, 2	1, 2	—
Gmean$_s$	1.966	2.759	3.345	4.069	2.862
	—	1	1	all	—
F_1 *score*	2.517	2.828	3.517	2.897	3.241
	—	—	1, 2	—	—
precision	3.034	2.897	3.379	2.138	3.552
	4	4	4	—	4
recall	2.000	2.707	3.345	4.276	2.672
	—	1	1, 2	all	—
specificity	3.931	3.103	3.069	1.483	3.414
	2, 3, 4	4	4	—	4
	ROS	SMOTE	SVM	B2	RSMO-Q3
	(1)	(2)	(3)	(4)	(5)
BAC	2.086	2.707	3.310	3.828	3.069
	—	—	1, 2	1, 2	—
Gmean$_s$	1.966	2.724	3.310	4.000	3.000
	—	1	1	1, 2, 3	—
F_1 *score*	2.552	2.828	3.483	2.862	3.276
	—	—	1, 2	—	—
precision	3.069	2.931	3.414	2.224	3.362
	4	4	4	—	4
recall	2.017	2.672	3.293	4.207	2.810
	—	1	1, 2	all	—
specificity	4.069	3.207	3.241	1.517	2.966
	all	4	4	—	4

Observations

From the obtained results, it can be concluded that the use of the *multistage majority voting organization* may allow, in the case of imbalanced data classification task, to improve the ensemble performance when compared to the traditional combination of the classifier pool. This is due to the division of classifiers into clusters containing models that make similar errors on problem instances. Thanks to this, after the first voting level, the predictions reflecting the expert knowledge of the base models in each of the recognized feature space regions are obtained.

The introduction of sampling with replacement in order to further diversify the ensembles during the first voting stage while reducing the number of models making similar decisions allows for the improvement of the achieved results. This method can be regarded as related to the static selection of classifiers. Both in the case of GNB and CART, it led to an increase in the ability of the proposed methods to detect minority class, which is particularly visible in the case of algorithms based on the averaged Q statistics.

Compared to classifier ensembles employing preprocessing techniques, the proposed methods are characterized by a lower ability to detect the minority class. It is worth noting, however, that only in a few cases these differences were statistically significant.

Answers to research questions

The answers to the previously formulated research questions are as follows:

Q1. Can the use of a clustering-based *multistage majority voting organization* improve the performance of the imbalanced data classification?

A1. The conducted experiments confirmed that the use of methods based on a *multistage majority voting organization* may lead to the improvement of the ensemble methods performance in the imbalanced data classification task.

Q2. Can the introduction of sampling with replacement before the first voting stage of TSMV allow to increase the ability to detect minority class?

A2. The obtained results confirmed that the addition of sampling with replacement to the the TSMV algorithm (RSMO) allows to improve the detection ability of minority class objects.

Q3. Can the proposed algorithms compete with methods employing preprocessing techniques?

A3. The results of the conducted research confirmed that in most cases the proposed TSMV and *rsmo* algorithms are not statistically significantly worse than ensemble methods using preprocessing techniques.

3.3 Distance-Based Dynamic Classifier Selection

This section proposes two *dynamic classifier selection* algorithms for the imbalanced data classification task. These are respectively the *Dynamic Ensemble Selection using Euclidean distance* (DESE) and the *Dynamic Ensemble Selection using Imbalance Ratio and Euclidean distance* (DESIRE). The introduction of these methods is motivated by the – indicated in the literature – shortage of *dynamic classifier selection* approaches dedicated to the task of unbalanced data classification [59]. Imbalanced learning continues to be an important problem in pattern recognition, especially in the case of real-world data. As the methods of dynamic selection of classifiers perform a local classification – based on the local area of competence often defined as the nearest neighborhood of the classified instance – they may allow reducing the bias in relation to the majority class. Nevertheless, there are currently very few DES algorithms dedicated for the problem of imbalanced data classification.

The generation of the classifier pool is based on the *Bagging* approach [30], and more specifically on the *Stratified Bagging*, in which the samples are drawn with replacement from the minority and majority class separately in such a way that each bootstrap maintains the original training set class proportion. This is necessary due to the high imbalance, which in the case of standard *Bagging* can lead to the generation of training sets containing only the majority class.

Both proposed methods are derived in part from algorithms based on local oracles, and more specifically on KNORA-U [123], which gives base classifiers weights based on the number of correctly classified instances in the local region of competence and then combines them by weighted majority voting. The computational cost in this type of method is mainly related to the size of the classifier pool and the $\mathcal{DSEL}$ size, as the *k-Nearest Neighbors* technique is used to define local competence regions, which can be costly for large datasets. Instead of voting, DESE and DESIRE are based on *support functions* and they calculate weights for each classifier for both the minority and majority classes separately. These weights are calculated on the basis of *the Euclidean distance* (L_2 norm) between the classified sample and its neighbors in the local region of competence. The literature indicates, with respect to the commonly used L_k norms, the potential usefulness of norms with the lower k value for problems with high dimensionality [3]. Examples of such metrics are *the Manhattan distance* (L_1 norm) or *a fractional distances*, in which case k may be less than 1. However, due to the relatively small dimensionality of the chosen datasets as well as popularity and frequent use in distance-based algorithms, *the Euclidean distance* was chosen as the base distance metric for the DESE and DESIRE.

Proposed methods come in two variants: *Positive* (denoted as P), where weights are modified only in the case of correct classification, and *Positive&Negative* (denoted as PN), where, in addition to correct decisions, weights are also affected by incorrect ones. The exact way of weights calculation is presented in Algorithm 5.

Algorithm 5 DESE and DESIRE weight calculation methods

```
Input:
    $\Pi$ – classifier pool,
    $\mathcal{TS}$ – testing set,
    $\mathcal{DSEL}$ – Dynamic Selection Dataset,
    $k$ – number of nearest neighbors,
    $min, maj$ – respectively the percentage of minority and majority classes in the training set,
    $W \leftarrow \varnothing$ – empty weights array of shape $(n, n, 2)$.
Symbols:
    $LRC_i$ – nearest neighborhood of sample $x_i$,
    $TP, FN$ – true positive and false negative,
    $n$ – number of base classifiers,
Output:
    $W$ – weights array of shape $(n, n, 2)$.

 1: for each sample $x_i$ in $\mathcal{TS}$ do
 2:     $LRC_i \leftarrow$ the $k$ nearest neighbors of $x_i$ in $\mathcal{DSEL}$
 3:     for each Classifier $\Psi_j$ in $\Pi$ do
 4:         $Predict \leftarrow$ PREDICT$(LRC_i, \Psi_j)$
 5:         for each $neighbor$ in $LRC_i$ do
 6:             if $Predict[neighbor] = TP$ then
 7:                 $W[j,i,0] += \begin{cases} \|x_i - neighbor\| & \text{for DESE} \\ \|x_i - neighbor\| * min & \text{for DESIRE} \end{cases}$
 8:             else if $Predict[neighbor] = TP$ then                                  Positive
 9:                 $W[j,i,1] += \begin{cases} \|x_i - neighbor\| & \text{for DESE} \\ \|x_i - neighbor\| * maj & \text{for DESIRE} \end{cases}$
10:             else if $Predict[neighbor] = FN$ then                                  Positive&Negative
11:                 $W[j,i,1] -= \begin{cases} \|x_i - neighbor\| & \text{for DESE} \\ \|x_i - neighbor\| * min & \text{for DESIRE} \end{cases}$
12:             else if $Predict[neighbor] = FN$ then
13:                 $W[j,i,0] -= \begin{cases} \|x_i - neighbor\| & \text{for DESE} \\ \|x_i - neighbor\| * maj & \text{for DESIRE} \end{cases}$
14:         end for
15:     end for
16: end for
```

For each instance, the proposed algorithms perform the following steps:

- In step 2, the *k-Nearest Neighbors* of a given instance x_i are found in $\mathcal{DSEL}$, which form the local region of competence LRC_i.
- In step 4, each classifier Ψ_j from the pool classifies all samples belonging to LRC_i.
- In steps 5-13, the classifier weights are modified separately for the minority and majority class, starting from the value of 0. The *Positive&Negative* variant uses

all four conditions, while the *Positive* variant is based only on the conditions in lines 6 and 8. In the case of DESE, the modifications are based on *the Euclidean distance* between the classified sample and its neighbor from the local competence region, and in the case of DESIRE, *the Euclidean distance* is additionally scaled by a percentage of the minority or majority class in such a way that more emphasis is placed on the minority class.

Finally, the weights obtained from DESE or DESIRE are normalized to the $[0, 1]$ range and multiplied by the ensemble support matrix. The combination is carried out according to the maximum rule [76], which chooses the classifier that is most confident of itself. This combination rule, despite its potentially sound grounds, is rarely used in practice due to its high sensitivity to overfitting. Using the most confident classifier may mean choosing an over-trained model whose generalization ability has been significantly impaired. However, if the dimensionality of the analyzed problem is relatively low, the possibility of overfitting is accordingly reduced. This is due to the potentially lower number of noisy features and low sparsity of the feature space.

Computational and memory complexity analysis

The proposed method for each sample $x_i \in \mathcal{TS}$ finds its local neighborhood in $\mathcal{DSEL}$ using the *k-Nearest Neighbors* algorithm. Each distance computation has the complexity of $O(d)$, where d is the problem's dimensionality. Distance is calculated from x_i to each instance in $\mathcal{DSEL}$ which results in $O(d \mid \mathcal{DSEL} \mid)$ runtime, where $\mid \mathcal{DSEL} \mid$ is a cardinality of $\mathcal{DSEL}$. Then, kNN selects k neighbors for each sample in $\mathcal{DSEL}$, which requires $O(\mid \mathcal{DSEL})$. This, in total, results in the computation complexity of $O(d \mid \mathcal{DSEL} \mid + k \mid \mathcal{DSEL} \mid))$.

Next, each classifier $\Psi_j \in \Pi$ labels k neighbors of x_i and based on the classification results uses the calculated distance to establish the weight for a given classifier. This step has the computational complexity of $O(nk)$.

3.3.1 Experimental evaluation

This subsection presents the motivation, goals and set-up of the performed experiments, as well as their results.

Research questions

The experiments were designed to answer the following questions:

Q1. Does taking into account *the Euclidean distance* to a given neighbor of a classified sample in the process of local competency estimation allow the algorithm to deal with the imbalanced data classification problem?

Q2. Does the introduction of the weighting of *the Euclidean distance* using the imbalance ratio in such a way as to put more emphasis on the minority class lead to an increase in the algorithm's ability to detect a given class?

Goals of the experiments

Experiment 1 – Euclidean distance-based approach
The main goal of the first experiments was to compare the performance of proposed dynamic selection methods, weighted based on Euclidean distance, with the *state-of-art* ensemble methods paired with preprocessing.

Experiment 2 – Scaled Euclidean distance-based approach
The aim of the second experiment was to check how the previously proposed method would behave after taking into account during weights calculation process the difference between the majority and minority classes.

Experimental set-up

The experiments were carried out on 41 imbalanced datasets presented in Table 2.3 on p. 44. The evaluation of the proposed methods is based on five metrics widely used in the case of imbalanced classification problems. Three popular classification algorithms were used as base models, and *Random Oversampling* was employed to investigate the impact of simple data preprocessing on the proposed ensemble methods. Classifier pools of four different sizes were generated using *Stratified Bagging*. As a reference method, a single classifier, as well as *Stratified Bagging* (SB) and dynamic selection in the form of the KNORA-U algorithm were selected. This choice is aimed at comparing the proposed methods with a combination of the entire classifier pool, as well as with the *state-of-the-art* dynamic selection method in the task of imbalanced data classification. Both proposed and reference methods occur in versions with preprocessing (in the form of *Random Oversampling*) and without it, the use of oversampling is denoted by the letter O added to the method's acronym. Detailed information is presented below:

- Evaluation measures – *balanced accuracy score* (BAC), *Gmean$_s$*, *F_1 score*, *precision*, and *recall*,

- Classification algorithms – *Gaussian Naïve Bayes* classifier (GNB), *k-Nearest Neighbors* classifier (kNN), and *Classification and Regression Tree* (CART),
- Data preprocessing – *Random Oversampling* (ROS).
- Classifier pool size – consecutively 5, 15, 30 and 50 base models,
- Reference methods – a single model (GNB\CART\$k$NN), *Stratified Bagging* (SB), *Stratified Bagging* with ROS (SBO), KNORA-U, and KNORA-U with ROS (KNORA-UO).

The evaluation was carried out using 10 times repeated 5-fold cross-validation. Due to the small number of instances in the datasets, $\mathcal{DSEL}$ is defined as the entire training set. All experiments have been implemented in *Python* and can be replicated using the code available on *Github*[3].

The radar diagrams show the average global ranks achieved by each of the tested algorithms in terms of each of the 5 evaluation metrics, while the tables show the results of the Wilcoxon rank-sum ($p = 0.05$) statistical test for a pool size of 5 base classifiers. The numbers under the average rank of each method indicate the algorithms which are statistically significantly worse than the one in question.

Experiment 1 – Euclidean distance-based approach

Figure 3.10 shows how the average ranks for DESE and the reference methods change with respect to different metrics as a function of ensemble size. The proposed methods (in particular DESE-PO) for 5 base models achieve higher ranks with respect to each metric with an exception of *recall*. While the single classifier and *bagging* prefer *recall*, DESE-PO and DESE-P *precision*. As the number of base classifiers increases, BAC and $Gmean_s$-based rankings deteriorate to KNORA-U levels, while F_1 *score* remains high due to high *precision*.

Table 3.8 presents the results of the statistical analysis, which shows that the DESE-PO method performs statistically significantly better than all reference methods with respect to every metric except *recall*.

When the base classifier is CART, as seen in Figure 3.11, for the smallest pool, DESE-P (both without and with oversampling) ranks higher than the reference methods with respect to each of the five metrics. As the number of base models increases, KNORA-UO and SBO stand out with respect to *precision*, DESE-PO performs better with respect

[3]`https://github.com/w4k2/iccs20-desire`

to other metrics, and DESE-PNO achieves the highest average ranks in terms of BAC, $Gmean_s$ and *recall* despite the low F_1 *score* and *precision*. Table 3.9 confirms that for the five basic classifiers, DESE-PO is statistically significantly better than all reference methods, while DESE-PNO performs statistically significantly better than DESE-PO with respect to *recall*, $Gmean_s$ and BAC.

Figure 3.12 and table 3.10 show that the proposed methods using oversampling are not statistically different from the reference methods, except for a single classifier that excels in *precision*, but at the same time achieves the worst mean ranks based on the remaining metrics. Together with the increase in the number of base classifiers, KNORA-U and SBO achieve higher mean ranks than DESE-PO and DESE-PNO.

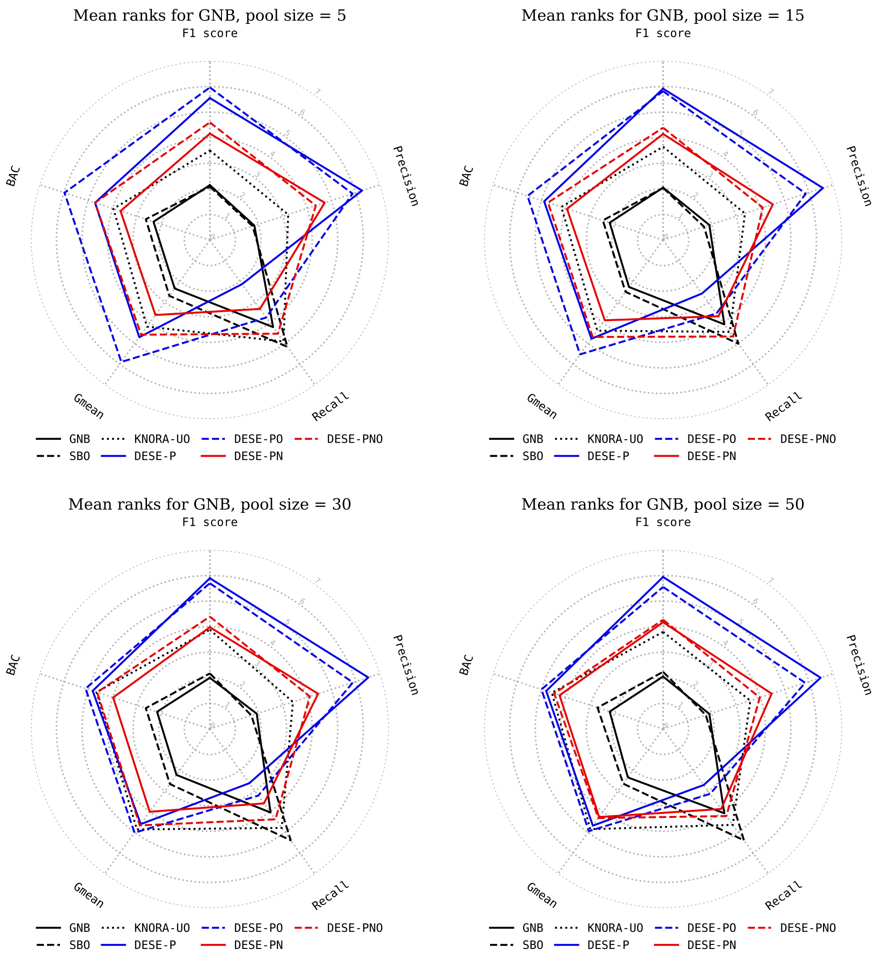


Figure 3.10: *Mean ranks for* GNB *classifier.*

Table 3.8: *Statistical tests on mean ranks for* GNB *with pool size* = *5. The higher the average rank value, the better.*

	GNB (1)	SBO (2)	KNORA-UO (3)	DESE-P (4)	DESE-PO (5)	DESE-PN (6)	DESE-PNO (7)
F_1 *score*	2.146	2.085	3.500	5.549	5.963	4.159	4.598
	–	–	1,2	1,2,3,6,7	1,2,3,6,7	1,2,3	1,2,3
precision	1.829	1.756	3.220	6.256	5.866	4.720	4.354
	–	–	1,2	*all*	1,2,3,6,7	1,2,3	1,2,3
recall	4.207	5.159	4.902	2.134	3.744	3.329	4.524
	4	4,5,6	4,5,6	–	4	4	4,5,6
Gmean$_s$	2.341	2.695	4.183	4.695	5.890	3.622	4.573
	–	–	1,2	1,2,6	*all*	1	1,2,6
BAC	2.317	2.634	3.963	4.720	5.976	3.671	4.720
	–	–	1,2	1,2,6	*all*	1,2	1,2,6

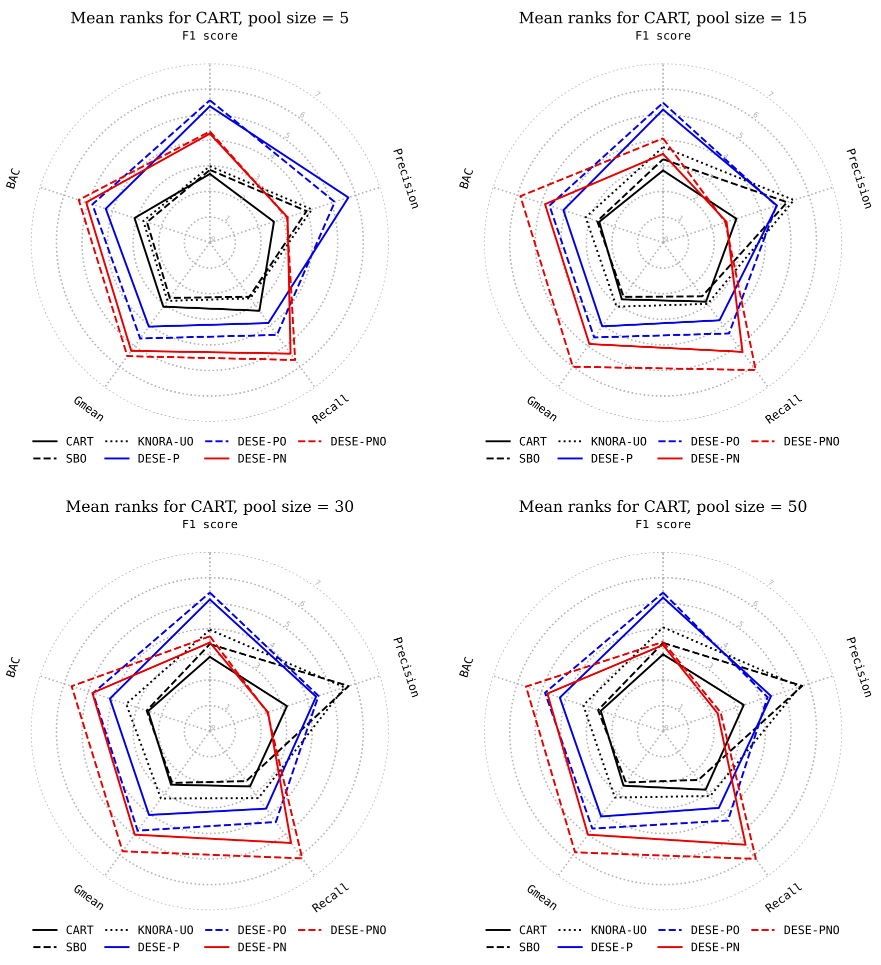


Figure 3.11: *Mean ranks for* CART *classifier.*

Table 3.9: *Statistical tests on mean ranks for* CART *with pool size = 5. The higher the average rank value, the better.*

	CART	SBO	KNORA-UO	DESE-P	DESE-PO	DESE-PN	DESE-PNO
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
F_1 *score*	2.683	2.841	2.988	5.329	5.561	4.256	4.341
	–	–	–	1,2,3,6,7	1,2,3,6,7	1,2,3	1,2,3
precision	2.634	3.976	4.195	5.695	5.134	3.195	3.171
	–	1	1,6,7	*all*	1,2,3,6,7	–	–
recall	3.293	2.622	2.695	3.890	4.463	5.366	5.671
	2,3	–	–	2,3	1,2,3,4	1,2,3,4,5	1,2,3,4,5
Gmean$_s$	3.098	2.671	2.817	4.061	4.634	5.232	5.488
	–	–	–	2,3	1,2,3,4	1,2,3,4	1,2,3,4,5
BAC	3.098	2.585	2.732	4.280	4.829	5.085	5.390
	–	–	–	1,2,3	1,2,3,4	1,2,3,4	1,2,3,4,5

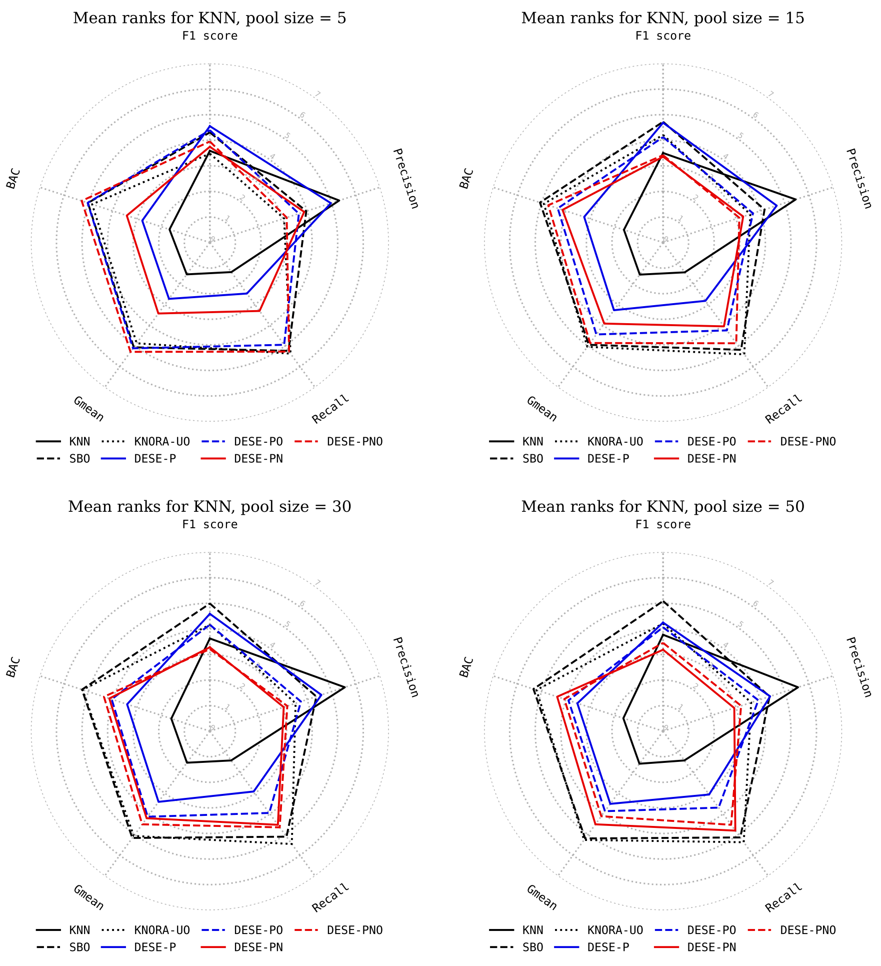


Figure 3.12: *Mean ranks for* kNN *classifier.*

Table 3.10: *Statistical tests on mean ranks for* kNN *with pool size = 5. The higher the average rank value, the better.*

	kNN (1)	SBO (2)	KNORA-UO (3)	DESE-P (4)	DESE-PO (5)	DESE-PN (6)	DESE-PNO (7)
F_1 score	3.585	4.305	3.476	4.549	4.390	3.744	3.951
	–	3	–	1,6	–	–	–
precision	5.317	3.963	3.049	4.976	3.659	3.878	3.159
	3,5,6,7	3,7	–	2,3,5,6,7	–	7	–
recall	1.427	5.232	5.366	2.463	4.939	3.305	5.268
	–	1,4,6	1,4,6	1	1,4,6	1,4	1,4,6
$Gmean_s$	1.537	5.061	4.866	2.720	5.110	3.427	5.280
	–	1,4,6	1,4,6	1	1,4,6	1,4	1,4,6
BAC	1.659	5.012	4.841	2.780	5.024	3.415	5.268
	–	1,4,6	1,4,6	1	1,4,6	1,4	1,4,6

Experiment 2 – Scaled Euclidean distance-based approach

The results in Figures 3.13–3.15 and Tables 3.11–3.13 show the average ranks for the proposed DESIRE method, which calculates weights based on *the Euclidean distances* scaled by the percentages of the minority and majority classes in the training set.

In the case of GNB as the base model (Figure 3.13), the DESIRE-PO method achieves the best results compared to reference methods in terms of mean ranks based on *F_1 score*, *precision*, *$Gmean_s$* and BAC. When the ensemble size increases, the proposed method is equal to KNORA-UO in terms of BAC and *$Gmean_s$* but retains the advantage in terms of *F_1 score* and *precision*. Moreover, the more base classifiers used, the smaller the differences between DESIRE with preprocessing and the version without preprocessing. Table 3.11 presents the results of the statistical analysis, which shows that DESIRE-PO is statistically better than all reference methods when the number of base classifiers is small.

Figure 3.14 shows that for a small classifier pool, DESIRE-PO achieves higher ranks than reference methods in terms of each evaluation metric, and as the classifier number increases, it loses significantly in *precision* compared to SBO and KNORA-UO. DESIRE-PNO has a high *recall*, which unfortunately is reflected by the lowest *precision* and *F_1 score*. Table 3.12 shows that for 5 base classifiers, DESIRE-C both with and without preprocessing is statistically significantly better than reference methods in terms of all metrics except one, *$Gmean_s$* in the case DESIRE-P and *recall* for DESIRE-PO.

When the base classifier is kNN (Figure 3.15), as in the case of DESE, DESIRE-PO is not statistically worse than SBO and KNORA-UO (Table 3.13) and as the number of classifiers in the pool increases, the average global ranks significantly deteriorate compared to reference methods.

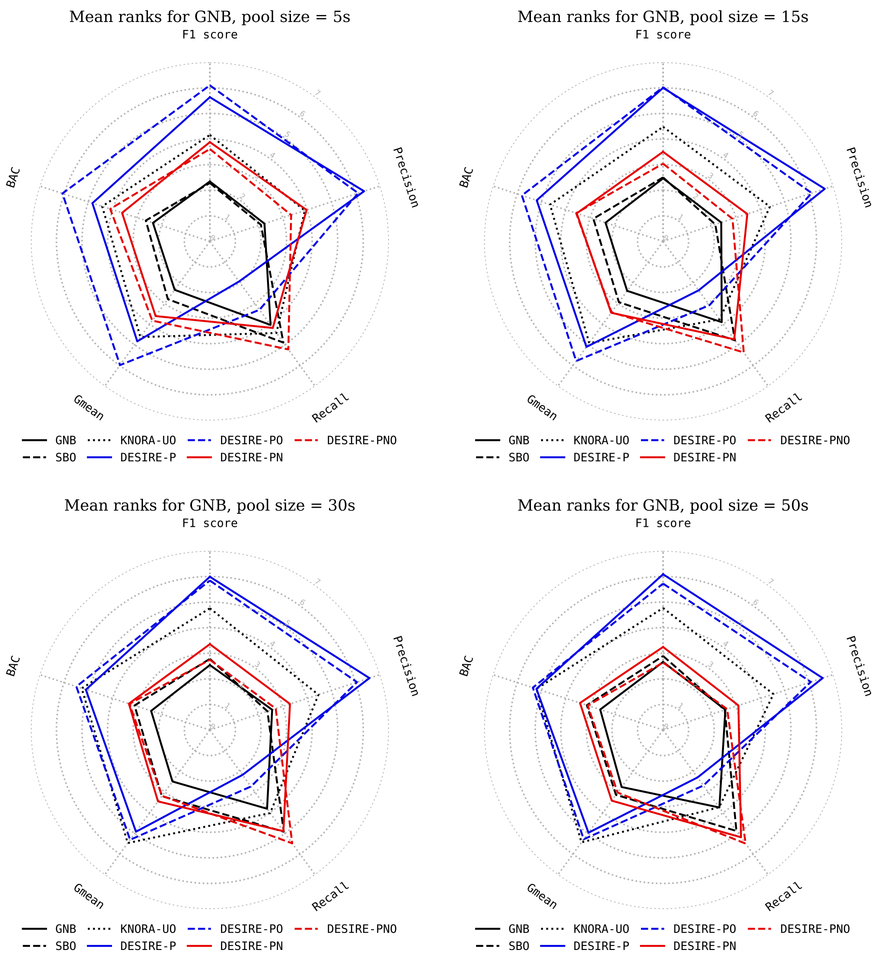


Figure 3.13: *Mean ranks for* GNB *classifier.*

Table 3.11: *Statistical tests on mean ranks for* GNB *with pool size = 5. The higher the average rank value, the better.*

	GNB (1)	SBO (2)	KNORA-UO (3)	DESIRE-P (4)	DESIRE-PO (5)	DESIRE-PN (6)	DESIRE-PNO (7)
F_1 *score*	2.341	2.280	4.159	5.634	6.098	3.878	3.610
	–	–	1,2	1,2,3,6,7	1,2,3,6,7	1,2	1,2
precision	2.244	2.098	3.902	6.341	6.098	3.976	3.341
	–	–	1,2	1,2,3,6,7	1,2,3,6,7	1,2,7	1,2
recall	4.037	4.890	4.427	1.939	3.305	4.183	5.220
	4	4,5	4,5	–	4	4,5	1,3,4,5,6
Gmean$_s$	2.341	2.793	4.622	4.829	5.976	3.610	3.829
	–	–	1,2,6	1,2,6,7	*all*	1	1,2
BAC	2.341	2.634	4.427	4.829	6.061	3.610	4.098
	–	–	1,2,6	1,2,6	*all*	1,2	1,2

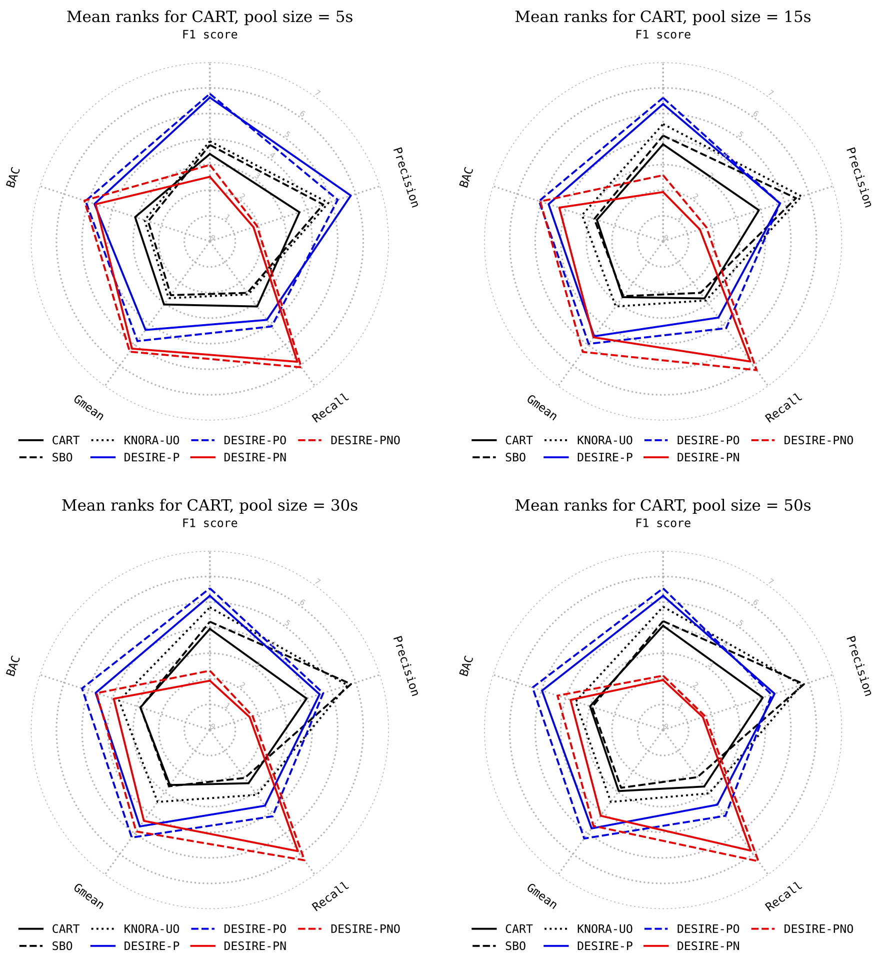


Figure 3.14: *Mean ranks for* CART *classifier.*

Table 3.12: *Statistical tests on mean ranks for* CART *with pool size = 5. The higher the average rank value, the better.*

	CART (1)	SBO (2)	KNORA-UO (3)	DESIRE-P (4)	DESIRE-PO (5)	DESIRE-PN (6)	DESIRE-PNO (7)
F_1 score	3.415	3.768	3.915	5.622	5.768	2.524	2.988
	6	6	6,7	1,2,3,6,7	1,2,3,6,7	–	–
precision	3.683	4.659	4.878	5.793	5.256	1.793	1.939
	6,7	1,6,7	1,6,7	*all*	1,6,7	–	–
recall	3.146	2.488	2.561	3.793	4.110	5.817	6.085
	2,3	–	–	2,3	1,2,3	1,2,3,4,5	1,2,3,4,5
$Gmean_s$	3.049	2.598	2.744	4.280	4.817	5.183	5.329
	–	–	–	1,2,3	1,2,3,4	1,2,3,4	1,2,3,4
BAC	3.073	2.537	2.683	4.744	5.110	4.695	5.159
	–	–	–	1,2,3	1,2,3	1,2,3	1,2,3

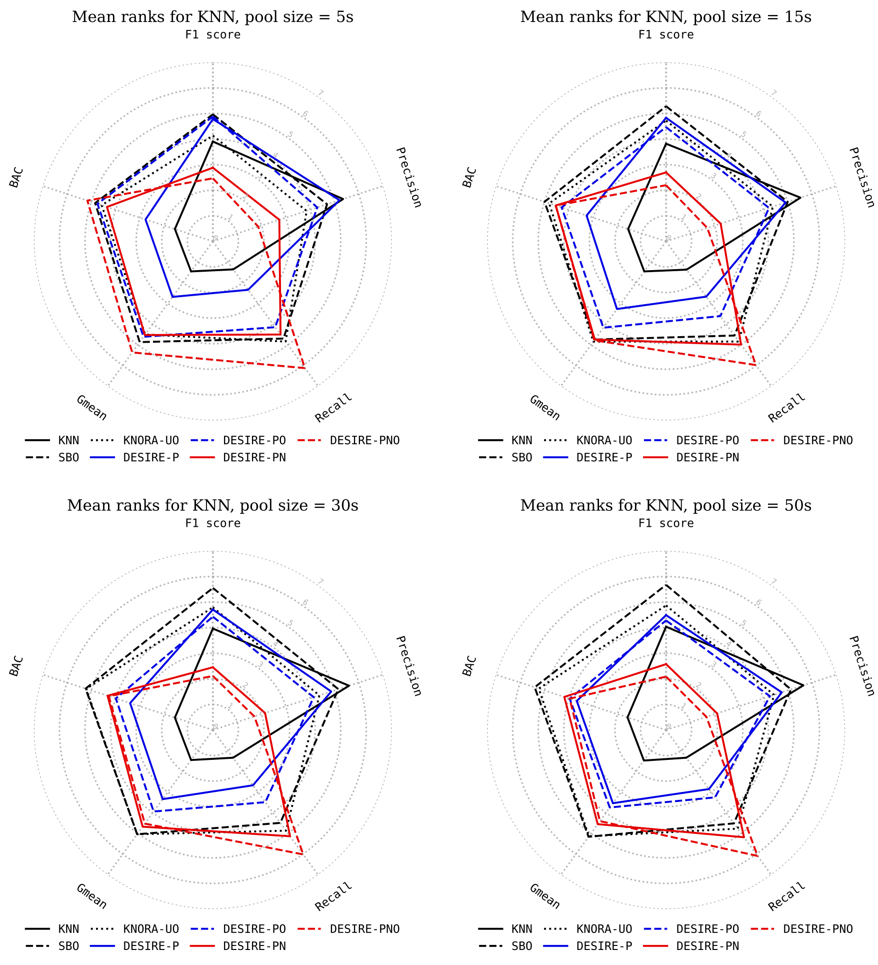


Figure 3.15: *Mean ranks for* *k*NN *classifier.*

Table 3.13: *Statistical tests on mean ranks for* *k*NN *with pool size* = *5. The higher the average rank value, the better.*

	*k*NN	SBO	KNORA-UO	DESIRE-P	DESIRE-PO	DESIRE-PN	DESIRE-PNO
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
F_1 *score*	3.902	4.963	4.134	4.780	4.878	2.878	2.463
	6,7	1,3,6,7	6,7	6,7	6,7	–	–
precision	5.354	4.695	3.854	5.207	4.293	2.732	1.866
	5,6,7	3,6,7	6,7	3,5,6,7	6,7	7	–
recall	1.354	4.695	4.841	2.341	4.146	4.500	6.122
	–	1,4	1,4	1	1,4	1,4	*all*
$Gmean_s$	1.451	4.866	4.500	2.683	4.610	4.524	5.366
	–	1,4	1,4	1	1,4	1,4	1,3,4,5,6
BAC	1.561	4.841	4.573	2.768	4.744	4.354	5.159
	–	1,4	1,4	1	1,4	1,4	1,4,6

Observations

The results presented confirm that dynamic selection methods specifically adapted for classifying imbalanced data can achieve statistically better results than ensemble methods coupled with preprocessing, especially when the pool of base classifiers is relatively small. This may be because *Bagging* has not yet stabilized while the proposed method selects the best single classifier. The *Positive* approach, in which the weights of the models were changed only when the instances belonging to the local competence region were correctly classified, proved to be more balanced with respect to all 5 evaluation measures. This could indicate excessive weight penalties for misclassification in the *Positive&Negative* approach. When *k*NN is used as the baseline classifier, the proposed methods performed statistically similar to KNORA-U for a small pool, and they ranked statistically worse compared to the reference methods for a larger number of classifiers. This is probably due to the method used to compute the support in the *k*NN, which is not suitable for the algorithms proposed in this work. For GNB and CART, DESE-P and DESIRE-P achieved results that are statistically better or similar to the reference methods, often without the use of preprocessing, since it has a built-in mechanism to handle the imbalance.

Answers to research questions

The answers to the previously formulated research questions are as follows:

Q1. Does taking into account *the Euclidean distance* to a given neighbor of a classified sample in the process of local competency estimation allow the proposed algorithm to deal with the imbalanced data classification problem?

A1. The obtained results confirmed that taking into account *the Euclidean distance* to a given neighbor of a classified sample in the process of local competency estimation may allow the proposed algorithm to deal with the imbalanced data classification problem.

Q2. Does the introduction of the weighting of *the Euclidean distance* using the imbalance ratio in such a way as to put more emphasis on the minority class lead to an increase in the algorithm's ability to detect a given class?

A2. The conducted experiments confirmed that, in the case of All variant, the introduction of the weighting based on imbalance ratio may lead to an increase in the algorithm's ability to detect a minority class instances.

Chapter 4

Algorithms for imbalanced data stream classification

This chapter is focusing on combining two of the important research topics associated with data analysis, i.e., data stream classification as well as data analysis with imbalanced class distributions. It introduces new algorithms designed specifically for these kinds of tasks, employing methods of *Dynamic Ensemble Selection.* Simultaneously introducing new ways to use DES algorithms in the imbalanced data stream classification.

First, the novel highly imbalanced data stream classification method, employing a classifier selection approach in order to focus on the detection of the minority class, which can update its model when new data arrives is proposed.

Next, two novel frameworks employing integrating data preprocessing and dynamic ensemble selection methods for imbalanced data stream classification are introduced. In the first case, single pattern recognition models are used as base classifiers, while the second approach employs *Stratified Bagging* for base classifier generation.

4.1 *Minority Driven Ensemble*

In this section, the algorithm *Minority Driven Ensemble* (MDE) is proposed to address the problem of classifying highly imbalanced data streams with *concept drift*. The proposed MDE method was intended to fill the gap in algorithms for classifying imbalanced data streams that was hinted at in Chapter 1. Many real-world data streams have high *Imbalance Ratio* , and existing methods dedicated to this problem often have high computational complexity. Therefore, the assumption in the design of MDE was to achieve relatively low computational complexity by using a simple approach to building and maintaining an ensemble of classifiers and the absence of data preprocessing techniques in the form of undersampling or oversampling. The ensemble construction in MDE is based on the SEA algorithm, and the prediction process uses a novel combination rule based on the notion of classifier selection. Therefore, the proposed method fits the approaches from *the inbuilt mechanism* group.

Ensemble construction

The proposed algorithm does not detect *a concept drift* occurrence, but instead employs a mechanism allowing it to construct self-adapting classifier ensemble. For each data stream chunk $\mathcal{DS}_k$, the *k-Nearest Neighbors* classifier is trained based on the data devoided of *outliers* according to 5-neighbor taxonomy [179] (i.e., samples from minority class for which five nearest neighbors are majority class examples).

If the fixed ensemble size n_{max} is exceeded, the worst rated individual classifier according to the *Balanced Accuracy Score* (BAC) is removed from the classifier pool Π. Additionally, at each step all models with BAC are lower than $0.5 + \alpha$, where α is the algorithm's parameter responsible for the outdated models removing rate, are removed from Π. The pseudocode of the presented method is shown in Algorithm 6. The description of the functions used in the training phase pseudocode is as follows:

- REMOVEOUTLIERS() – removes the *outliers* from the current data chunk $\mathcal{DS}_k$ according to 5-neighbor taxonomy.
- TRAIN() – builds new classifier Ψ_k on the current data chunk $\mathcal{DS}_k$.
- EVALUATE() – calculates the *balance accuracy score* on current data chunk $\mathcal{DS}_k$ for each base classifier $\Psi_i \in \Pi$ in order to use it later in the pruning process.
- PRUNETHRESHOLD() – removes from pool Π all models with BAC lower than α.
- PRUNEWORST() – removes from pool Π the model with the lowest BAC.

Algorithm 6 Training phase of the MDE algorithm

```
Input:
    $Stream = \{\mathcal{DS}_1, \mathcal{DS}_2, \ldots, \mathcal{DS}_k, \mathcal{DS}_{k+1}, \ldots\}$ – data stream,
    $n_{max}$ – maximal number of base models,
    $\alpha$ – outdated models removing rate,
Symbols:
    $\Pi$ – classifier pool,
    $\mathcal{S}_k$ – set of evaluation metric values for each base classifier,

1: $\Pi \leftarrow \varnothing$
2: for each $k, \mathcal{DS}_k = \{x_k^1, x_k^2, \ldots, x_k^N\}$ in $Stream$ do
3:     $\mathcal{S}_k \leftarrow \varnothing$
4:     if $k == 0$ then
5:         $\mathcal{DS}_k \leftarrow$ REMOVEOUTLIERS($\mathcal{DS}_k$)
6:         $\Psi_k \leftarrow$ TRAIN($\mathcal{DS}_k$)
7:         $\Pi \leftarrow \Psi_k$
8:     else
9:         $\mathcal{S}_k =$ EVALUATE($\Pi, \mathcal{DS}_k$)
10:        if $\mid \Pi \mid > 1$ then
11:            $\Pi \leftarrow$ PRUNETHRESHOLD($\Pi, \mathcal{S}_k, \alpha$)
12:        if $\mid \Pi \mid > n_{max} - 1$ then
13:            $\Pi \leftarrow$ PRUNEWORST($\Pi, \mathcal{S}_k$)
14:        $\mathcal{DS}_k \leftarrow$ REMOVEOUTLIERS($\mathcal{DS}_k$)
15:        $\Psi_k \leftarrow$ TRAIN($\mathcal{DS}_k$)
16:        $\Pi \leftarrow \Psi_k$
17: end for
```

Prediction

During the prediction process if at least one individual classifier returns a non-zero support for minority class – i.e., among k nearest neighbors, at least one belongs to minority class – then the instance is classified as the minority class example.

The concept of the proposed combination rule is presented in Figure 4.1. The first three subplots present the decision border implementing the principle of minimum support for three subsequent processed data chunks during subtle changes in the minority class distribution. The last subplot (on the right) shows the illustration of the mentioned above combination rule.

Algorithm 7 Prediction phase of the MDE algorithm

Input:

$Stream = \{\mathcal{DS}_1, \mathcal{DS}_2, \ldots, \mathcal{DS}_k, \mathcal{DS}_{k+1}, \ldots\}$ – data stream,

$\Pi = \{\Psi_1, \Psi_2, \ldots, \Psi_n\}$ – classifier pool,

Output:

$Decision$ – classification results.

```
1: for each $k, \mathcal{DS}_k = \{x_k^1, x_k^2, \ldots, x_k^N\}$ in $Stream$ do
2:     $esm_k$ = ENSEMBLESUPPORTMATRIX($\Pi, \mathcal{DS}_k$)
3:     $ms_k$ = MAJORITYSUPPORT($esm_k$)
4:     $mms_k$ = MINMAJORITYSUPPORT($ms_k$)
5:     $Decision$ = INT($mms_k$)          ▷ If support is less than 100% then 0, otherwise 1
6: end for
```

The description of the functions used in the prediction phase pseudocode is as follows:

- ENSEMBLESUPPORTMATRIX() – returns an array of shape $(\|\Pi\|, N, 2)$ containing base classifiers' supports for each of N samples in a given da chunk $\mathcal{DS}_k$,
- MAJORITYSUPPORT() – returns only the majority class support from esm_k,
- MINMAJORITYSUPPORT() – returns the minimum of ms_k,
- INT() – return an integer object constructed from given values of minimal majority support mms_k.

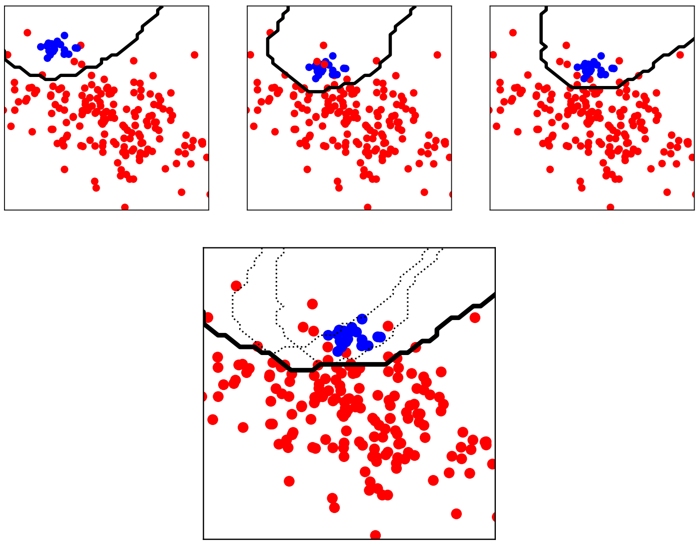


Figure 4.1: *Binary prediction as non-zero support for a minority class (three on the top) and a maximum from the pool (on the bottom).*

Computational and memory complexity analysis

Both the removal of outliers and the classification process are performed using the *k-Nearest Neighbors* based on the *Euclidean distance*. Each distance computation has the complexity of $O(d)$, where d is the problem's dimensionality. Distance is calculated from each classified instance in $\mathcal{DS}_k$ to all samples used to train a given kNN classifier Ψ_j, which results in $O(dN)$ runtime, where N is a cardinality of each data chunk. Then, kNN selects k neighbors for each sample in $\mathcal{DS}_k$, which requires $O(kN)$. This, in total, results in the computation complexity of $O(dN + kN)$.

During the prediction process, for each of N problem instances in a given data chunk $\mathcal{DS}_k$, MDE calculates the minimal majority support in order to find a model with a non-zero support for minority class. This operation is a modification of *support accumulation* combination rule and has a computational complexity of $O(n)$.

4.1.1 Experimental evaluation

This subsection presents the motivation, goals and set-up of the performed experiments, as well as their results.

Research questions

The experiments were designed to answer the following questions:

Q1. Can the use of the proposed strategy based on non-zero support for a minority class lead to better results in the case of highly imbalanced data stream than those obtained by classical *Dynamic Ensemble Selection* algorithms?

Q2. Is the proposed method, based largely on the neighborhood defined by the kNN classifier, resistant to label noise and *concept drift* occurrence?

Goals of the experiments

Experiment 1 – Hyperparameters optimization

The main goal of the first experiment was to tune the two hyperparameters of MDE:

- n_{max} — ensemble size,
- α — pruning parameter responsible for the outdated models removing rate.

Both mean BAC values and statistical dependence for multiple values of these two parameters were reported.

Experiment 2 – Comparative analysis of classifier selection methods

During the second experiment, the performance of MDE was compared to the four reference *Dynamic Selection* (DS) techniques implemented in *DESlib* [57]. The comparison was made in terms of *Imbalance Ratio* value, *concept drift* type and the level of label noise.

Experimental set-up

The experiments were carried out based on 96 diverse data streams generated using the *stream-learn* [141] package. Each of the streams contains the total of 100 000 instances, divided into 200 chunks of 500 objects described by 8 features, and contains 5 *concept drifts*. The variety of generated data streams was obtained by generating 3 replication of each combination of the following parameters:

- *the imbalance ratio* — successively 10, 20, 30 and 40% of the minority class,
- *the level of label noise* — successively 0, 10, 20 and 30%,
- *the type of concept drift* — *gradual* or *sudden.*

Additionally, during Experiment 2, the proposed method was evaluated on the 5 real data streams described in Table 4.1.

Table 4.1: *Real data streams characteristics.*

Data stream	#Samples	#Features	IR
covtypeNorm-1-2vsAll	266 000	54	4
poker-lsn-1-2vsAll	360 000	10	10
INSECTS-abrupt_imbalanced_norm	300 000	33	19
INSECTS-gradual_imbalanced_norm	100 000	33	19
INSECTS-incremental_imbalanced_norm	380 000	33	19

The evaluation of MDE is based on six metrics widely used in the case of imbalanced classification problems. As a reference methods, two *Dynamic Ensemble Selection* and two *Dynamic Classifier Selection* algorithms were selected. The number of nearest neighbors k used to define the local area of competence for *Dynamic Selection* methods was set at 7. This choice is aimed at comparing the proposed MDE methods with the *state-of-the-art Dynamic Selection* approaches in the task of imbalanced data stream classification. Detailed set-up is presented below:

- Evaluation measures – *balanced accuracy score* (BAC), *Gmean$_s$*, *F_1 score*, *precision*, *recall*, and *specificity*,

- Reference methods:
  - DES – KNORA-*Eliminate (*KNORA-E*)* and KNORA-*Union (*KNORA-U*)*,
  - DCS – *Modified Classifier Ranking (Rank)* and *Local classifier accuracy (*LCA*)*.

The evaluation was carried out using *Test-Then-Train* protocol. The *dynamic selection dataset* ($\mathcal{DSEL}$) for the DS methods was defined as the previous data chunk with the *Random Oversampling* performed on it. Conducted experiments as well as the MDE algorithm were implemented in *Python* programming language and may be repeated according to source code published on *Github*[1].

Experiment 1 – Hyperparameters optimization

The following experiment was performed on the data stream with an *Imbalance Ratio* of 1 : 9 and 1% global label noise. *Sudden* and *gradual concept drifts* were tested separately. The results of hyperparameter optimization are shown in Figure 4.2, which shows the relationship between the parameter α (X-axis) and the ensemble size (Y-axis). Each value corresponds to the mean BAC obtained from MDE for given values of n_{max} and α. The colors correspond to the statistical dependencies between the mean BAC values, according to the *Wilcoxon rank-sum* test.

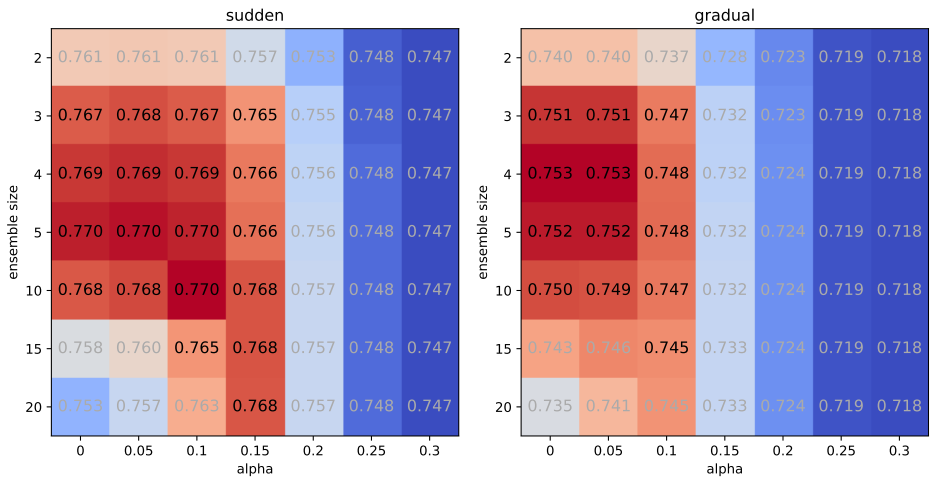


Figure 4.2: *Optimization of* MDE *hyperparameters for sudden and gradual concept drift in relation to the* Balanced Accuracy Score.

Increasing the size of the ensemble initially stabilizes the BAC, but over time degrades the ability of the ensemble to respond to the *concept drift*. Increasing the removal rate α parameter initially compensates for the degradation of the *concept drift* response time, but at the same time negatively affects the BAC value.

The $n_{max} = 3$ and the $\alpha = 0.05$ were chosen for further experiments.

[1]`https://github.com/w4k2/classifier-selection`

Experiment 2 – Comparative analysis of classifier selection methods

Figure 4.3 shows the influence of random over-sampling on reference methods performance on data streams with high *Imbalance Ratio* (1 : 9). The use of oversampling equates the performance of all tested DS methods.

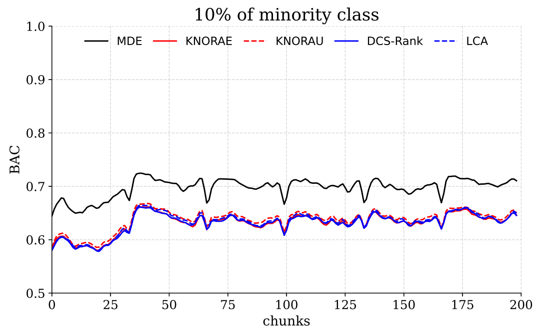


Figure 4.3: *Reference methods performance with (left) and without oversampling (right).*

Figure 4.4 shows how the performance of the methods depends on the *Imbalance Ratio*. The proposed MDE is very effective for highly imbalanced data streams (10%, 20% of minority class samples). Increasing the percentage of minority class to 30% reduces the differences between MDE and the reference methods. In the cases of low imbalance data (40% of minority class), MDE performs worse than the reference *Dynamic Selection* methods.

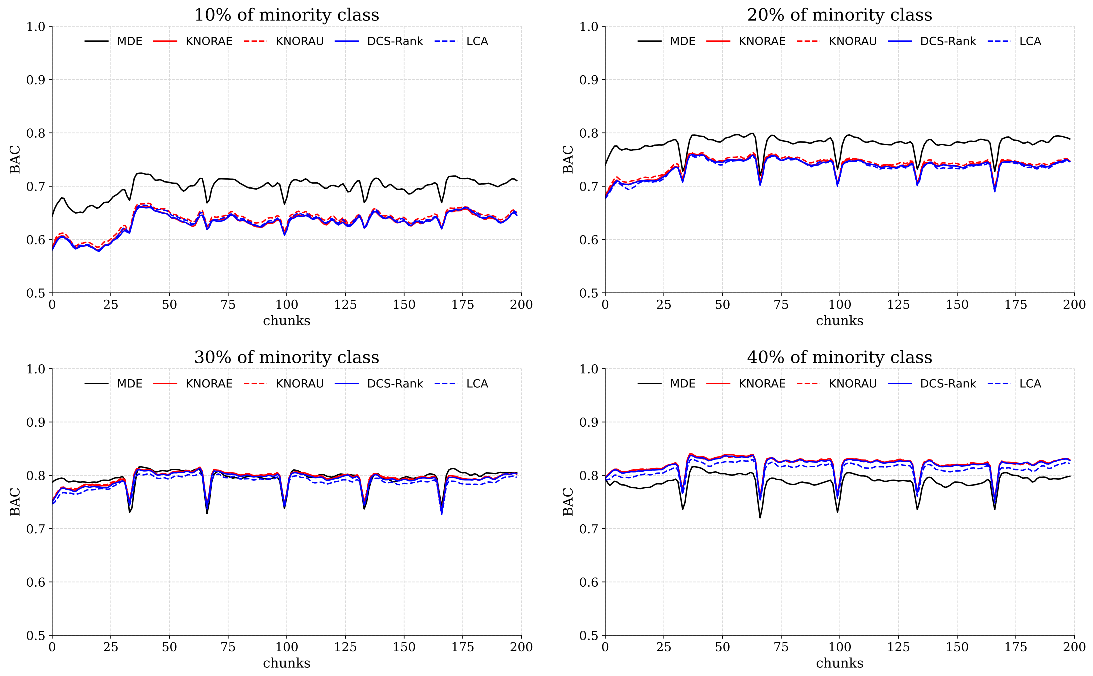


Figure 4.4: *Influence of imbalance scale on the quality of classification.*

The aim of the experiment is to demonstrate the ability of the proposed method to classify highly imbalanced data, so all further results are presented for streams with a percentage of a minority class not greater than 20%.

Figure 4.5 presents the relation between the classification quality and the type of *concept drift*. The type of the *concept drift* does not affect the relation between the analyzed classification methods. In either case MDE outperforms the benchmark classifiers.

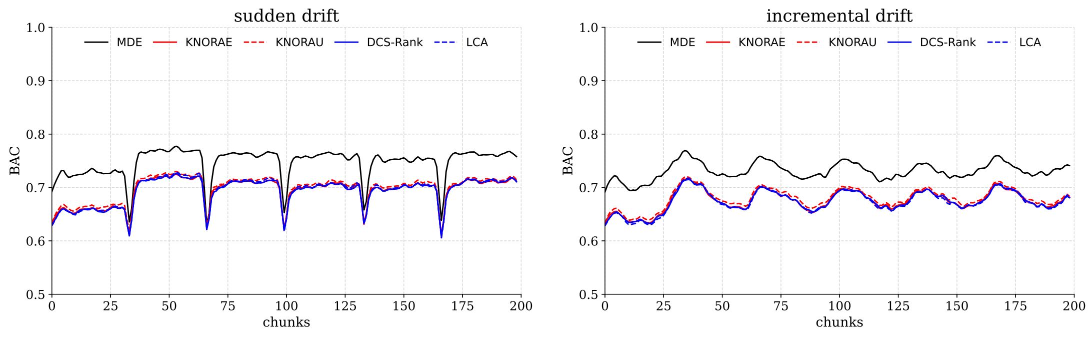


Figure 4.5: *Influence of concept drift type on the quality of classification.*

Figure 4.6 shows the relation between the performances of the individual methods and the label noise ratio. The increase of noise has a negative effect on the overall generalization ability.

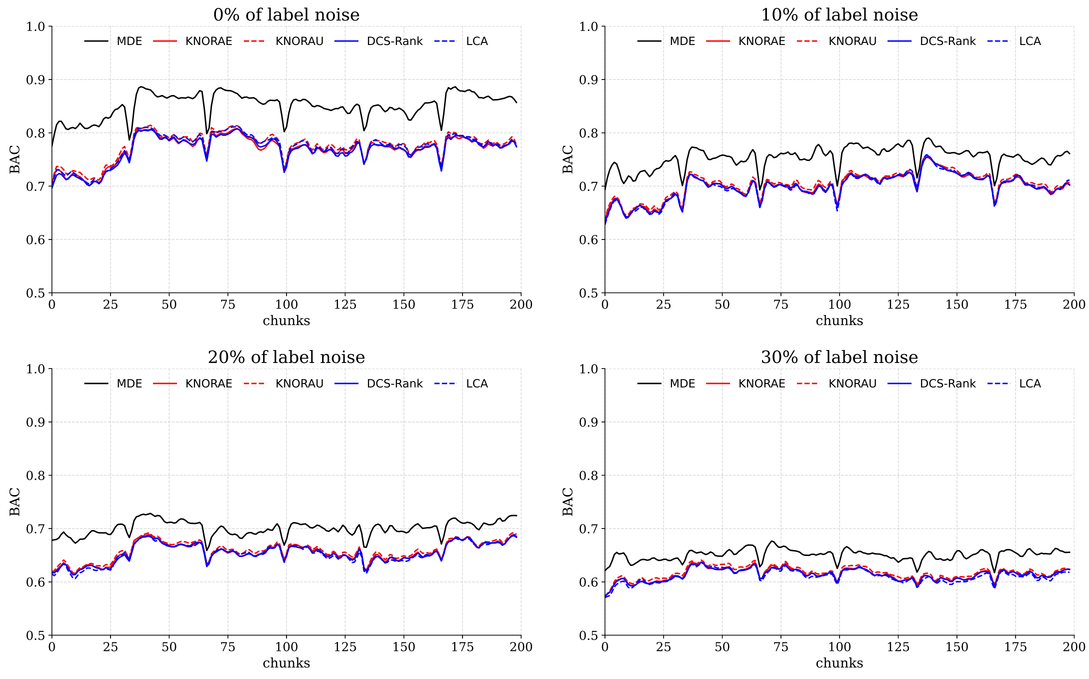


Figure 4.6: *Influence of label noise on the quality of classification.*

The statistical analysis of the experimental evaluation is presented in Table 4.2. It confirms that MDE performs better than the benchmark classifier selection methods in most cases. Only for slightly imbalanced data, i.e., when *Imbalance Ratio* is small (30% of minority examples), MDE is not statistically significantly better than KNORAU and KNORAE. For nearly balanced data streams (40% of minority examples), RANK, KNORAU, and KNORAE are better than MDE.

Table 4.2: *Presentation of statistical dependency of methods in all analyzed contexts. Bold points the highest* BAC *value for a given context.*

value	MDE	K-E	K-U	Rank	LCA
		Minority class percentage			
10%	**0.697**	0.632	0.637	0.631	0.634
20%	**0.780**	0.738	0.741	0.736	0.735
30%	**0.796**	**0.794**	**0.794**	0.792	0.786
40%	0.788	**0.821**	**0.821**	**0.820**	0.811
		Drift types			
incremental	**0.731**	0.675	0.680	0.675	0.674
sudden	**0.747**	0.694	0.698	0.693	0.694
		Label noise			
0%	**0.851**	0.770	0.776	0.769	0.773
10%	**0.753**	0.700	0.704	0.699	0.699
20%	**0.701**	0.656	0.659	0.655	0.654
30%	**0.651**	0.614	0.617	0.613	0.611

Additionally, Figure 4.7 shows the results achieved by MDE in comparison with reference methods for the task of the real imbalanced data stream classification. Radar charts show the averaged values of the evaluation metrics achieved by each method, while the runs depict balanced accuracy values over the entire length of the data stream.

It is worth noting that in the case of the covtypeNorm stream, which is characterized by the lowest *Imbalance Ratio* among all real data streams, MDE achieves the results at the level of the reference methods and, additionally, does not display a visible decrease presented by the reference methods at the end of the presented run. There is also a decrease in the *precision* value at the expense of a slight increase in *recall*, which indicates that the method prefers the minority class.

In the case of a poker stream, where the *Imbalance Ratio* is higher, the potential of the proposed method can be seen. Despite the precision value at the level of the reference methods, the MDE presents a much better ability to detect minority class at the cost of a decrease in *specificity*. Additionally, the presented method achieves a much higher *Gmean*$_s$ value, and a slightly better F_1 *score*, and *Balanced Accuracy Score*. The presented run shows that MDE, in the case of poker stream, can achieve up to 80% *Balanced Accuracy Score*.

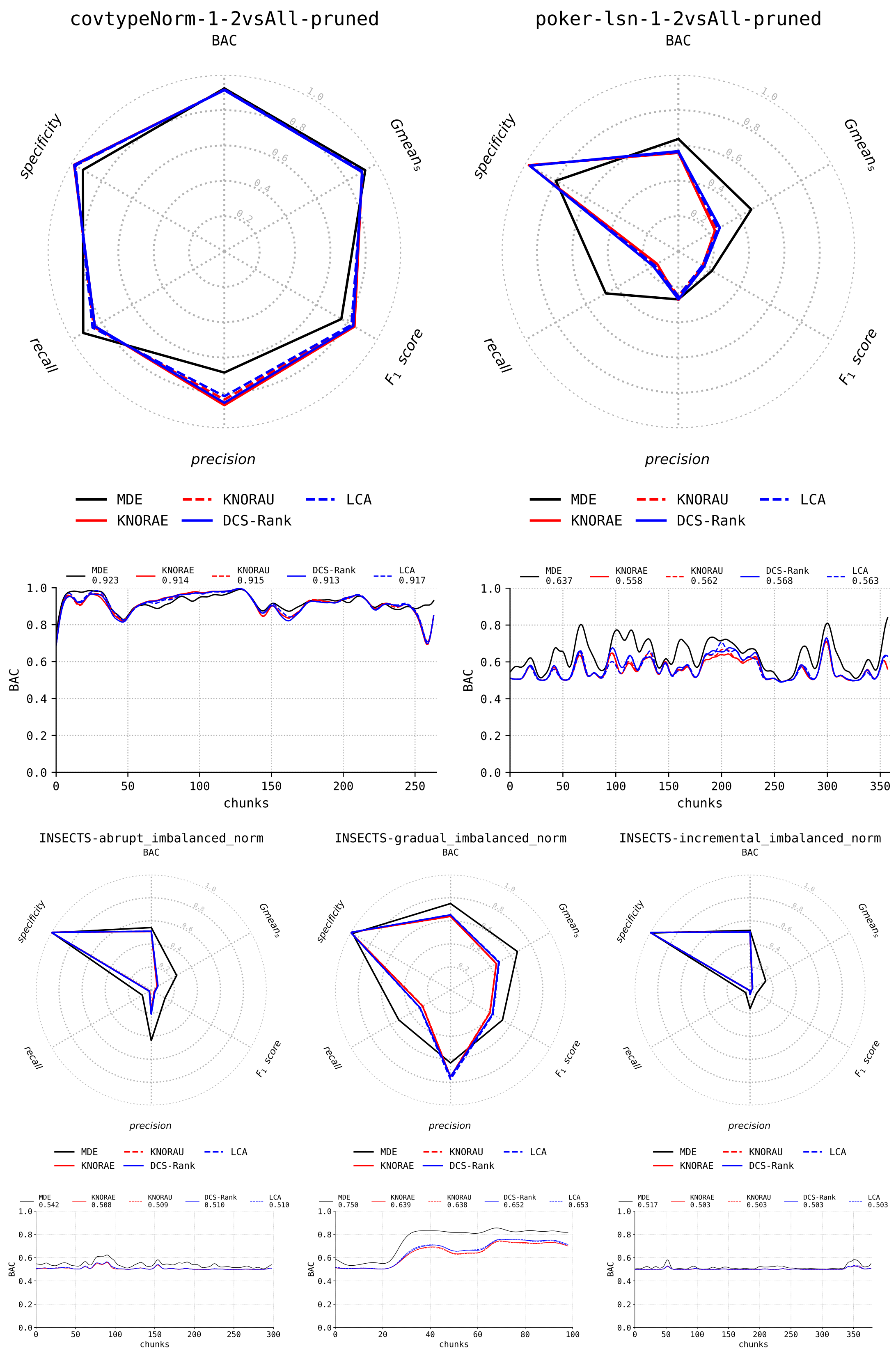


Figure 4.7: *Results of the MDE comparison with reference methods for real data streams.*

The results achieved on the INSECTS streams, which display the highest imbalance, are also interesting. In the case of the stream with *gradual concept drift*, it can be seen that the proposed method achieves significantly higher results in terms of BAC, $Gmean_s$, F_1 *score*, and *recall*, with the *specificity* value equal to the reference methods. All this is achieved at the cost of a slight decrease in *precision*. The presented run shows that MDE maintains a high value of *Balanced Accuracy Score* along the entire length of the stream.

The potential of the method is visible especially in the results obtained on INSECTS streams containing sudden and *incremental concept drift*. Despite their difficulty and the fact that the reference methods achieve results close to the random classifier, MDE is able to break out of this minimum at times, showing its potential to deal with even extremely difficult problems.

Observations

Based on the conducted experiments, it can be assert that, especially for highly imbalanced data streams, MDE is statistically significantly better that *state-of-the-art* classifier selection methods. Additionally, MDE is quite robust to label noise and does not allow for significant deterioration of its classification performance in the case of concept drift appearance. It is also worth noting that the behavior displayed by MDE on synthetic streams was confirmed in experiments using real data streams. In their case, MDE also showed the potential to deal with highly imbalanced problems. Interestingly, in the case of the *covtypeNorm* stream, the generalizing ability of the proposed method did not seem to deteriorate, as in the case of synthetic streams with a lower *Imbalance Ratio*.

Answers to research questions

The answers to the previously formulated research questions are as follows:

Q1. Can the use of the proposed strategy based on non-zero support for a minority class lead to better results in the case of highly imbalanced data stream than those obtained by classical *Dynamic Ensemble Selection* algorithms?

A1. The obtained results confirmed that the MDE algorithm may outperform the *state-of-the-art Dynamic Selection* methods in the task of highly imbalanced data stream classification.

Q2. Is the proposed method, based largely on the neighborhood defined by the kNN classifier, resistant to label noise and *concept drift* occurrence?

A2. The conducted experiments confirmed the resistance of the MDE to both global label noise and *concept drift* occurrence.

4.2 *Dynamic Ensemble Selection* for Imbalanced Stream Classification

This section introduces the *Dynamic Ensemble Selection for Imbalanced Stream Classification* (DESISC) framework for the task of drifting imbalanced data stream classification. The ensemble's construction is based on the *Streaming Ensemble Algorithm* (SEA) concept [221], with an additional threshold-based pruning, and various *oversampling* techniques are used to deal with class imbalance. The motivation for this proposal was, among others, the shortage of methods dedicated to the imbalanced data stream classification stream, presented in Chapter 1. An additional goal was to propose a novel use of *Dynamic Ensemble Selection* in combination with preprocessing for imbalanced classification, which so far has been considered in the literature rarely and only for static data [198]. By using *Dynamic Selection*, taking into account the local competencies of the base classifiers, DESISC has a chance to deal not only with imbalance but also with the *concept drift* phenomenon, even without the use of preprocessing techniques.

DESISC framework

Each based model Ψ_k learns from the $\mathcal{LS}_k$ training set which is obtained by preprocessing $\mathcal{DS}_k$. $\mathcal{DSEL}_k$ denotes *dynamic selection dataset* for the kth data chunk and it is considered as previously preprocessed $\mathcal{DS}_{k-1}$. Each new trained classifier (one per each data chunk) is added to the ensemble until the maximum ensemble size n_{max} is achieved. Then if new model is added, each classifier in the ensemble is evaluated (according to BAC) and the worst one is removed. Additionally, at each step, all models which BAC scores are lower than a given threshold α are removed from the ensemble. Pruning process is performed before adding kth classifier to the pool. The concept behind the proposed framework is presented in Figure 4.8 and the pseudocodes for training and prediction phase is shown in Algorithms 8 and 9.

In the pseudocodes, the following functions were used:

- PREPROCESS() – generates the learning set $\mathcal{LS}_k$ by applying the chosen preprocessing method to the kth data chunk,
- TRAIN() – builds a new base classifier Ψ_k on the learning set $\mathcal{LS}_k$ generated by applying preprocessing to the kth data chunk,
- EVALUATE() – calculates the *balance accuracy score* on current data chunk $\mathcal{DS}_k$ for each base classifier $\Psi_i \in \Pi$ in order to use it later in the pruning process.
- PRUNETHRESHOLD() – removes from pool Π all models with BAC lower than α,

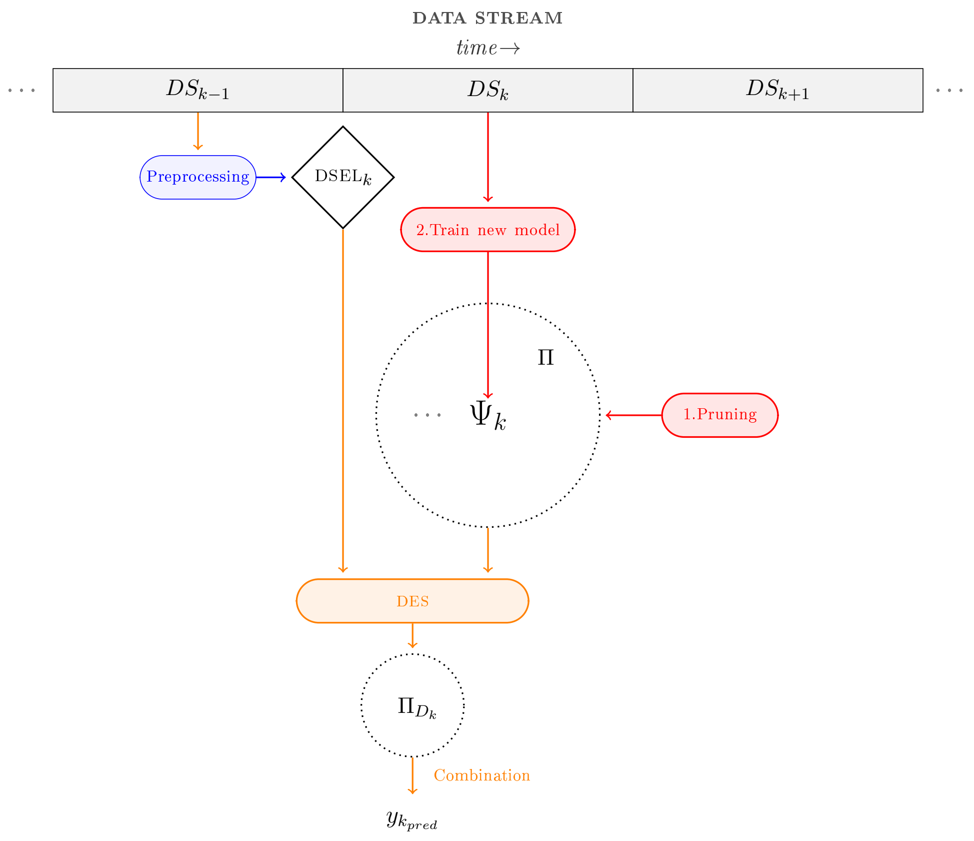


Figure 4.8: *The framework for training base classifiers and to prepare a* DSEL *for dynamic selection process. Here, $\mathcal{LS}_k$ is the learning set produced by preprocessing data chunk DS_k and Ψ_k is the base classifier trained on the kth data chunk. Π denotes the classifier pool.*

- PRUNEWORST() – – removes the worst-performing base classifier from the pool Π if the fixed maximum classifier pool size (n_{max}) is exceeded after adding a new model,
- PREDICT() – uses a given classifier pool (or list of ensembles in case of dynamic selection) to classify each instance in given data chunk,
- DYNAMICSELECTION() – uses a given dynamic ensemble selection method to generate a list of ensembles for classifying each test instance. In this work *Dynamic Ensemble Selection* can be performed on two levels - bagging classifier level or base estimators level.

In the beginning the classifier pool Π is empty. The first classifier Ψ_0 is generated using the preprocessed zero chunk (Algorithm 8 steps 4, 5 and 6). When the first data chunk arrives, the Ψ_0 is used to classify it. Then, the learning set $\mathcal{LS}_1$ is stored as the $\mathcal{DSEL}$ for the *Dynamic Ensemble Selection* process performed when next chunk arrives (Algorithm 9 step 5 and 6). $\mathcal{LS}_1$ is also used to train second base model (Algorithm 8 steps 4, 5 and 6). Then, with the arrival of each chunk, the following steps are performed:

Algorithm 8 Training phase of the DESISC framework

Input:

Stream – data stream,
n_{max} – maximum fixed size of the classifier pool,
α – pruning threshold,

Symbols:

$\mathcal{S}_k$ – set of evaluation metric values for each base classifier,
$\mathcal{DS}_k$ – data chunk,
Ψ_k – bagging classifier,
Π – bagging classifiers pool.

```
1: $\Pi \leftarrow \varnothing$
2: for each $k, \mathcal{DS}_k = \{x_k^1, x_k^2, \ldots, x_k^N\}$ in $Stream$ do
3:     if $k <= 1$ then                                         ▷ First data chunk.
4:         $\mathcal{LS}_k$ = PREPROCESS($\mathcal{DS}_k$)
5:         $\Psi_k \leftarrow$ TRAIN($\mathcal{LS}_k$)                     ▷ Bagging classifier generation
6:         $\Pi \leftarrow \Psi_k$                          ▷ Adding bagging classifier to the pool
7:     else                                     ▷ Third and all subsequent data chunks.
8:         $\mathcal{S}_k$ = EVALUATE($\Pi, \mathcal{DS}_k$)
9:         if $| \Pi |> 1$ then              ▷ Removing worst classifier if $n_{max}$ is exceeded.
10:            $\Pi \leftarrow$ PRUNETHRESHOLD($\Pi, \mathcal{S}_k, \alpha$)
11:        if $| \Pi |> n_{max} - 1$ then    ▷ Removing worst classifier if $n_{max}$ is exceeded.
12:            $\Pi \leftarrow$ PRUNEWORST($\Pi, \mathcal{S}_k$)
13:        $\mathcal{LS}_k$ = PREPROCESS($\mathcal{DS}_k$)
14:        $\Psi_k \leftarrow$ TRAIN($\mathcal{DS}_k$)
15:        $\Pi \leftarrow \Psi_k$
16: end for
```

Algorithm 9 Prediction phase of the DESISC frameworks

Input:

Stream – data stream,
Π – pool of bagging classifiers.

Symbols:

DS_k – data chunk,
Π_{D_k} – classifier ensemble selected using dynamic selection,
$\mathcal{DSEL}_k$ – dynamic ensemble selection dataset for the *k*th data chunk.

```
1: for each $k, \mathcal{DS}_k = \{x_k^1, x_k^2, \ldots, x_k^N\}$ in $Stream$ do
2:     if $k == 0$ then                                          ▷ First data chunk
3:         $Pass$                                                 ▷ No prediction
4:     else if $k == 1$ then                                    ▷ Second data chunk
5:         $Decision \leftarrow$ PREDICT($\mathcal{DS}_k, \Pi$)         ▷ Prediction using the whole pool
6:         $\mathcal{DSEL}_{k+1} \leftarrow$ PREPROCESS($\mathcal{DS}_k$)      ▷ Storing $\mathcal{DSEL}$ for next step
7:     else                                      ▷ Third and all subsequent data chunks
8:         $\Pi_{D_k} \leftarrow$ DYNAMICSELECTION($\Pi, \mathcal{DSEL}_k, \mathcal{DS}_k$)     ▷ Dynamic selection
9:         $Decision \leftarrow$ PREDICT($\mathcal{DS}_k, \Pi_{D_k}$)      ▷ Prediction using selected pool
10:        $\mathcal{DSEL}_{k+1} \leftarrow$ PREPROCESS($\mathcal{DS}_k$)
11: end for
```

1. Previously stored learning set is used as $\mathcal{DSEL}$ for the dynamic selection process to create the list of ensembles for classifying each instance in $\mathcal{DS}_k$ (Algorithm 9 step 8).
2. The ensembles selected by the chosen DES method are used to classify all instances int he current data chunk (Algorithm 9 step 9).
3. The current data chunk $\mathcal{DS}_k$ is preprocessed and stored as $\mathcal{DSEL}$ for the next *Dynamic Ensemble Selection* process (Algorithm 9 step 10).
4. All base models in classifier pool Π are evaluated based on BAC in order to use this information for ensemble pruning (Algorithm 8 step 8).
5. All base classifiers with BAC lower than a given threshold α are removed the the ensemble (Algorithm 8 steps 9 and 10).
6. The worst performing classifiers is removed from the ensemble is the maximal pool size n_{max} is exceeded (Algorithm 8 steps 11 and 12).
7. Using preprocessing, the learning set $\mathcal{LS}_k$ is generated, on the basis of which a new classifier is build and then added to the pool Π (Algorithm 8 steps 13, 14 and 15).

Computational and memory complexity analysis

Because the assumption of limited resources is crucial for the data stream processing, then let us estimate the computational complexity of the proposed framework. The proposed chunk-based framework for the imbalanced data stream classification is based on the methods of dynamic classifier selection as well as on preprocessing techniques (both *oversampling* and *undersampling*). For this reason, the key factors affecting the computational complexity of the presented approaches are, respectively, the number of models in the classifier pool for dynamic selection methods and the number of problem instances in a single data chunk in the case of preprocessing techniques.

Based on preliminary observations, it was established that the *Dynamic Ensemble Selection* methods (both KNORA-U and KNORA-E) have a linear time complexity of $O(n)$ depending on the number of base classifiers in the pool. The preprocessing techniques used in the work have, respectively, the logarithmic complexity of $O(log\ n)$ (ROS and RUS), the quadratic complexity of $O(n^2)$ (*Borderline2*-SMOTE) [260], and the complexity of $O(n\ log\ n)$ (CNN). SMOTE has the computational complexity of $O(n\ log_2\ n)$ [260]. Additionally DES-KNN performs calculation of pairwise disagreement measure ($O(n^2)$), and DES-CL employs the *k-means* clustering algorithm. The *k-means* computational complexity is $O(ncde)$, where c is the number of clusters, d is the number of data dimensions, and e describes the number of iterations/epochs [26]. Complexity is reduced to $O(nce)$, as the clustering space is one-dimensional.

Because a fixed size of the data chunk N is always set, the complexity of the proposed algorithms depends only on the number of classifiers from which the selection is made (denoted as $\mid \Pi \mid$).

4.2.1 Experimental evaluation

Here, the motivation, goals and set-up of the performed experiments are presented.

Research questions

The experiments were designed to answer the following questions:

Q1. Can the use of Dynamic Selection, taking into account the local competencies of the base classifiers, improve the ensemble's performance in the case of the imbalanced data stream with *concept drift*?

Q2. Can combining DES with data preprocessing improve the ensemble's performance in the case of the imbalanced data stream with *concept drift*?

Q3. Which DES methods and preprocessing techniques are best suited for the classification of a data stream with a given *concept drift* type and *Imbalance Ratio*?

Goals of the experiments

Experiment 1 – Imbalance Ratio impact

The aim of the first experiment is to test how DSEISC, with different combinations of *Dynamic Ensemble Selection* methods and preprocessing techniques, behaves when classifying data streams with various *Imbalance Ratios*.

Experiment 2 – Concept drift type impact

The aim of the first experiment is to evaluate how DSEISC, with different combinations of *Dynamic Ensemble Selection* methods and preprocessing techniques, behaves when classifying data streams with various types of *concept drift*.

Experimental set-up

The proposed framework was evaluated using 72 artificially generated data streams. Each stream is composed of one hundred thousand instances divided into 200 chunks of

500 objects described by 8 features, and contains 5 concept drifts. The base concepts were generated using the *stream-learn* package. The variety of streams was ensured by generating 3 replications with different random seed for each combination of the following parameters:

- *the imbalance ratio* — successively 10, 20, 30 and 40% of the minority class.
- *the level of label noise* — successively 0, 10 and 20%.
- *the type of concept drift* — *sudden* or *incremental.*

As the experimental protocol, the *Test-Than-Train* framework [135] was used, i.e., every classification model is trained on a recent data chunk, but it is evaluated on the basis of the following one. Evaluation of the DESISC was based on œetrics typical for imbalanced data classification problem. The value of pruning threshold α was set to .55, i.e., all base classifiers which BAC lower than .55 were removed from ensemble. This value was chosen in order to leave in the classifier pool only the models that performed slightly better than the random classifier. The maximum size of the classifier pool n_{max} was set to 20. Neighborhood size for *Dynamic Ensemble Selection* methods was $k = 7$. Set-up details are listed below:

- Evaluation measures – *Balanced Accuracy Score* (BAC) and *Geometric mean score* ($Gmean_s$),
- Base classifier – *Classification and Regression Tree* (CART),
- Dynamic Selection Methods – KNORA-*Eliminate* (KNORA-E), KNORA-*Union* (KNORA-U), DES-kNN, DES-*Clustering* (DES-CL),
- Data preprocessing techniques – SMOTE, SVM-SMOTE, *Bordeline*-SMOTE in two variants (B1-SMOTE & B2-SMOTE), *Safe-level* SMOTE (SL-SMOTE), and ADASYN,
- Reference method – DESISC without DES and preprocessing, leaving a classifier pool combined using *support accumulation* (SACC).

Experiments were implemented in *Python* programming language and may be repeated according to source code published on *Github*[2].

[2]`https://github.com/w4k2/ECML19-IoT-DES-preproc`

Experiment 1 – *Imbalance Ratio* impact

The results of Experiment 1 according to BAC (a) and $Gmean_s$ (b) for different IR values are presented in Tables 4.3 and 4.4 and in Figures 4.9-4.12. Bold indicates the statistically significant best combination method, while brackets indicate the statistically significant best preprocessing algorithm for a given combination strategy. Small numbers below the results indicate the indices of methods that are statistically significantly outperformed by the considered combination strategy (best in row), while small letters represent preprocessing methods that are statistically significantly outperformed by the considered one (best in column). Statistical analysis was performed using the *Wilcoxon Signed Rank Test* ($p \leq .05$). The radar charts show how each data preprocessing technique affected the performance of a particular *Dynamic Ensemble Selection* method, and are followed by the classification results for the best performing *Dynamic Selection* methods in conjunction with the most effective data preprocessing techniques. The methods presented were selected based on statistical evaluation and are compared with the *support accumulation* of the entire classifier pool and with the results obtained using only *Dynamic Ensemble Selection* or preprocessing.

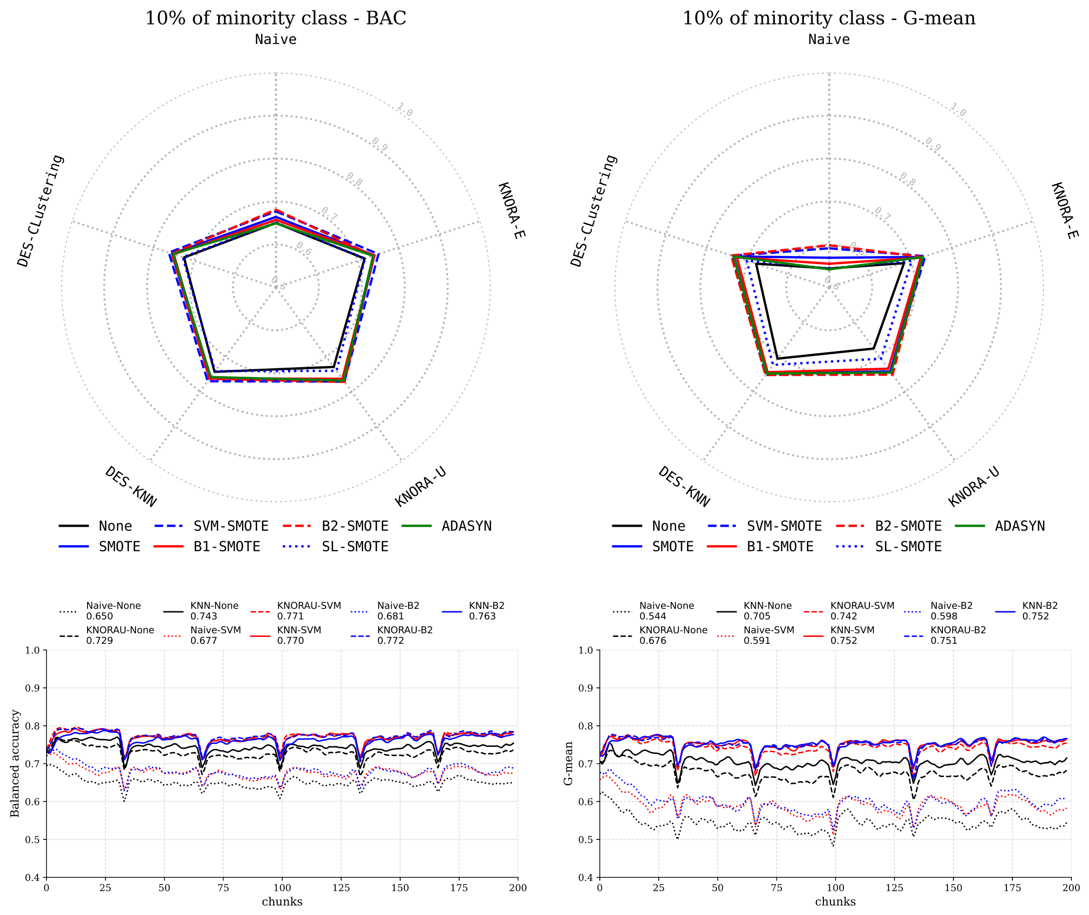


Figure 4.9: *Comparison of different sampling approaches for different classifier ensembles with respect to performance measures (*BAC *and* $Gmean_s$*) for imbalance ratio* 1 : 9.

Table 4.3: *Results of the Wilcoxon Signed Rank Test for various Imbalance Ratios in relation to* BAC.

1:9 IR	SACC (1)	KNORA-E (2)	KNORA-U (3)	DES-kNN (4)	DES-Cl (5)
None ($_a$)	0.650 – g	0.717 1 –	0.729 1,2,5 –	**0.743** All f	0.725 1,2 –
SMOTE ($_b$)	0.664 – a,d,f,g	0.741 1 a,e,f,g	**0.768** All a,d,f	0.762 1,2,5 a,f,g	0.754 1,2 a,d,f,g
SVM-SMOTE ($_c$)	0.677 – a,b,d,f,g	[0.751] 1 All	**0.771** All a,b,d,f,g	[0.770] 1,2,5 All	[0.762] 1,2 All
B1-SMOTE ($_d$)	0.657 – a,f,g	0.741 1 a,e,f,g	**0.763** All a,f	0.762 1,2,5 a,f,g	0.750 1,2 a,f
B2-SMOTE ($_e$)	[0.681] – All	0.738 1 a,f	**[0.772]** All All	0.763 1,2,5 a,b,f,g	0.755 1,2 a,b,d,f,g
SL-SMOTE ($_f$)	0.651 – g	0.718 1 –	**0.740** 1,2,5 a	**0.741** 1,2,5 –	0.728 1,2 a
ADASYN ($_g$)	0.649 – –	0.738 1 a,f	**0.768** All a,d,f	0.758 1,2,5 a,f	0.752 1,2 a,d,f

2:8 IR	SACC (1)	KNORA-E (2)	KNORA-U (3)	DES-kNN (4)	DES-Cl (5)
None ($_a$)	0.744 – –	0.779 1 f	0.809 1,2,5 –	**0.814** all f	0.800 1,2 –
SMOTE ($_b$)	0.757 – a,d,f,g	0.793 1 a,e,f,g	**0.829** All a,f	0.820 1,2,5 a,e,f,g	0.815 1,2 a,d,e,f,g
SVM-SMOTE ($_c$)	0.771 – a,b,d,f,g	[0.801] 1 All	**[0.833]** All All	[0.826] 1,2,5 All	[0.820] 1,2 All
B1-SMOTE ($_d$)	0.754 – a,f,g	0.793 1 a,e,f,g	**0.829** All a,f	0.820 1,2,5 a,b,e,f,g	0.813 1,2 a,e,f
B2-SMOTE ($_e$)	[0.773] – All	0.782 1 a,f	**0.830** All a,b,d,f,g	0.814 1,2,5 f	0.811 1,2 a,f
SL-SMOTE ($_f$)	0.747 – a,g	0.776 1 –	**0.819** All a	0.805 1,2,5 –	0.800 1,2 a
ADASYN ($_g$)	0.744 – –	0.788 1 a,e,f	**0.830** All a,b,d,f	0.814 1,2 f	0.813 1,2 a,e,f

3:7 IR	SACC (1)	KNORA-E (2)	KNORA-U (3)	DES-kNN (4)	DES-Cl (5)
None ($_a$)	0.800 – –	0.806 1 e,f	**0.846** All –	0.844 1,2,5 e,f,g	0.834 1,2 f
SMOTE ($_b$)	0.806 – a,f,g	0.815 1 a,d,e,f,g	**0.856** All a,f	0.846 1,2,5 a,d,e,f,g	0.841 1,2 a,d,e,f,g
SVM-SMOTE ($_c$)	0.816 – a,b,d,f,g	[0.819] – All	**[0.858]** All All	[0.847] 1,2,5 All	[0.843] 1,2 All
B1-SMOTE ($_d$)	0.808 – a,b,f,g	0.813 1 a,e,f,g	**0.856** All a,b,e,f	0.844 1,2,5 e,f,g	0.839 1,2 a,e,f
B2-SMOTE ($_e$)	[0.819] 2 All	0.800 – –	**0.855** All a,f	0.836 1,2 f	0.835 1,2 a,f
SL-SMOTE ($_f$)	0.802 2 a,g	0.801 – –	**0.850** All a	0.833 1,2,5 –	0.831 1,2 –
ADASYN ($_g$)	0.800 – –	0.809 1 a,e,f	**0.856** All a,b,e,f	0.838 1,2 e,f	0.839 1,2,4 a,e,f

4:6 IR	SACC (1)	KNORA-E (2)	KNORA-U (3)	DES-kNN (4)	DES-Cl (5)
None ($_a$)	0.827 2 –	0.819 – e,f,g	**0.864** All –	[0.857] 1,2,5 All	0.851 1,2 e,f
SMOTE ($_b$)	0.828 2 a,f,g	0.823 – a,d,e,f,g	**0.867** All a,f	0.856 1,2,5 c,d,e,f,g	0.853 1,2 a,d,e,f,g
SVM-SMOTE ($_c$)	0.834 2 a,b,d,f,g	[0.823] – All	**[0.868]** All All	0.856 1,2,5 d,e,f,g	[0.853] 1,2 All
B1-SMOTE ($_d$)	0.832 2 a,b,f,g	0.821 – a,e,f,g	**0.867** All a,b,e,f	0.854 1,2,5 e,f,g	0.852 1,2 a,e,f
B2-SMOTE ($_e$)	[0.836] 2 All	0.811 – –	**0.866** All a,f	0.848 1,2 –	0.848 1,2 f
SL-SMOTE ($_f$)	0.827 2 –	0.815 – e	**0.864** All –	0.849 1,2,5 e	0.847 1,2 –
ADASYN ($_g$)	0.827 2 –	0.818 – e,f	**0.868** All a,b,e,f	0.852 1,2 e,f	0.852 1,2 a,e,f

Table 4.4: *Results of the Wilcoxon Signed Rank Test for various Imbalance Ratios in relation to $Gmean_s$.*

1:9 IR	SACC (1)	KNORA-E (2)	KNORA-U (3)	DES-kNN (4)	DES-Cl (5)
None ($_a$)	0.544	0.683	0.676	**0.705**	0.679
	–	1,3,5	1	*All*	1,3
	g	–	–	–	–
SMOTE ($_b$)	0.569	0.729	0.742	**0.748**	0.733
	–	1	1,2,5	*All*	1,2
	a,d,f,g	*a,d,f*	*a,d,f*	*a,d,f*	*a,d,f*
SVM-SMOTE ($_c$)	0.591	[0.735]	0.742	**[0.752]**	0.738
	–	1	1,2,5	*All*	1,2
	a,b,d,f,g	*All*	*a,d,f*	*a,b,d,f,g*	*a,b,d,f,g*
B1-SMOTE ($_d$)	0.555	0.724	0.734	**0.744**	0.726
	–	1	1,2,5	*All*	1,2
	a,f,g	*a,f*	*a,f*	*a,f*	*a,f*
B2-SMOTE ($_e$)	[0.598]	0.729	**[0.751]**	**[0.752]**	[0.740]
	–	1	1,2,5	1,2,5	1,2
	All	*a,d,f*	*All*	*a,b,d,f,g*	*All*
SL-SMOTE ($_f$)	0.544	0.702	0.705	**0.723**	0.702
	–	1	1,2,5	*All*	1
	g	*a*	*a*	*a*	*a*
ADASYN ($_g$)	0.542	0.729	0.745	**0.748**	0.734
	–	1	1,2,5	*All*	1,2
	–	*a,d,f*	*a,b,c,d,f*	*a,d,f*	*a,d,f*

2:8 IR	SACC (1)	KNORA-E (2)	KNORA-U (3)	DES-kNN (4)	DES-Cl (5)
None ($_a$)	0.704	0.768	0.792	**0.803**	0.783
	–	1	1,2,5	*All*	1,2
	–	–	–	*f*	–
SMOTE ($_b$)	0.724	0.789	**0.820**	0.816	0.807
	–	1	*All*	1,2,5	1,2
	a,d,f,g	*a,e,f,g*	*a,f*	*a,e,f,g*	*a,d,f*
SVM-SMOTE ($_c$)	0.744	[0.797]	**[0.825]**	[0.822]	[0.813]
	–	1	*All*	1,2,5	1,2
	a,b,d,f,g	*All*	*a,b,d,f,g*	*All*	*All*
B1-SMOTE ($_d$)	0.719	0.789	**0.821**	0.817	0.805
	–	1	*All*	1,2,5	1,2
	a,f,g	*a,e,f,g*	*a,b,f*	*a,b,e,f,g*	*a,f*
B2-SMOTE ($_e$)	[0.746]	0.780	**[0.825]**	0.812	0.806
	–	1	*All*	1,2,5	1,2
	All	*a,f*	*a,b,d,f,g*	*a,f*	*a,d,f*
SL-SMOTE ($_f$)	0.708	0.772	**0.809**	0.802	0.792
	–	1	*All*	1,2,5	1,2
	a,g	*a*	*a*	–	*a*
ADASYN ($_g$)	0.704	0.786	**0.822**	0.811	0.807
	–	1	*All*	1,2,5	1,2
	–	*a,e,f*	*a,b,d,f*	*a,f*	*a,d,f*

3:7 IR	SACC (1)	KNORA-E (2)	KNORA-U (3)	DES-kNN (4)	DES-Cl (5)
None ($_a$)	0.786	0.803	0.840	**0.841**	0.828
	–	1	1,2,5	*All*	1,2
	–	*e,f*	–	*e,f,g*	–
SMOTE ($_b$)	0.794	0.814	**0.852**	0.844	0.838
	–	1	*All*	1,2,5	1,2
	a,f,g	*a,d,e,f,g*	*a,f*	*a,d,e,f,g*	*a,d,e,f,g*
SVM-SMOTE ($_c$)	0.807	[0.817]	**[0.855]**	[0.846]	[0.841]
	–	1	*All*	1,2,5	1,2
	a,b,d,f,g	*All*	*All*	*All*	*All*
B1-SMOTE ($_d$)	0.797	0.811	**0.853**	0.843	0.836
	–	1	*All*	1,2,5	1,2
	a,b,f,g	*a,e,f,g*	*a,b,f*	*a,e,f,g*	*a,e,f*
B2-SMOTE ($_e$)	[0.810]	0.799	**0.853**	0.835	0.833
	2	–	*All*	1,2,5	1,2
	All	–	*a,b,f*	*f*	*a,f*
SL-SMOTE ($_f$)	0.790	0.799	**0.847**	0.831	0.828
	–	1	*All*	1,2,5	1,2
	a,g	–	*a*	–	–
ADASYN ($_g$)	0.786	0.808	**0.853**	0.837	0.836
	–	1	*All*	1,2	1,2
	–	*a,e,f*	*a,b,f*	*e,f*	*a,e,f*

4:6 IR	SACC (1)	KNORA-E (2)	KNORA-U (3)	DES-kNN (4)	DES-Cl (5)
None ($_a$)	0.822	0.818	**0.862**	[0.856]	0.849
	2	–	*All*	1,2,5	1,2
	–	*e,f*	–	*All*	*e,f*
SMOTE ($_b$)	0.824	0.822	**0.865**	0.855	0.851
	2	–	*All*	1,2,5	1,2
	a,f,g	*a,d,e,f,g*	*a,f*	*c,d,e,f,g*	*a,d,e,f*
SVM-SMOTE ($_c$)	0.831	[0.823]	**[0.867]**	0.854	[0.852]
	2	–	*All*	1,2,5	1,2
	a,b,d,f,g	*All*	*All*	*d,e,f,g*	*All*
B1-SMOTE ($_d$)	0.828	0.820	**0.866**	0.853	0.850
	2	–	*All*	1,2,5	1,2
	a,b,f,g	*a,e,f,g*	*a,b,e,f*	*e,f,g*	*a,e,f*
B2-SMOTE ($_e$)	[0.833]	0.810	**0.865**	0.847	0.847
	2	–	*All*	1,2	1,2,4
	All	–	*a,f*	–	*f*
SL-SMOTE ($_f$)	0.822	0.814	**0.863**	0.848	0.845
	2	–	*All*	1,2,5	1,2
	–	*e*	*a*	*e*	–
ADASYN ($_g$)	0.822	0.817	**0.866**	0.851	0.850
	2	–	*All*	1,2	1,2
	–	*e,f*	*a,b,e,f*	*e,f*	*a,e,f*

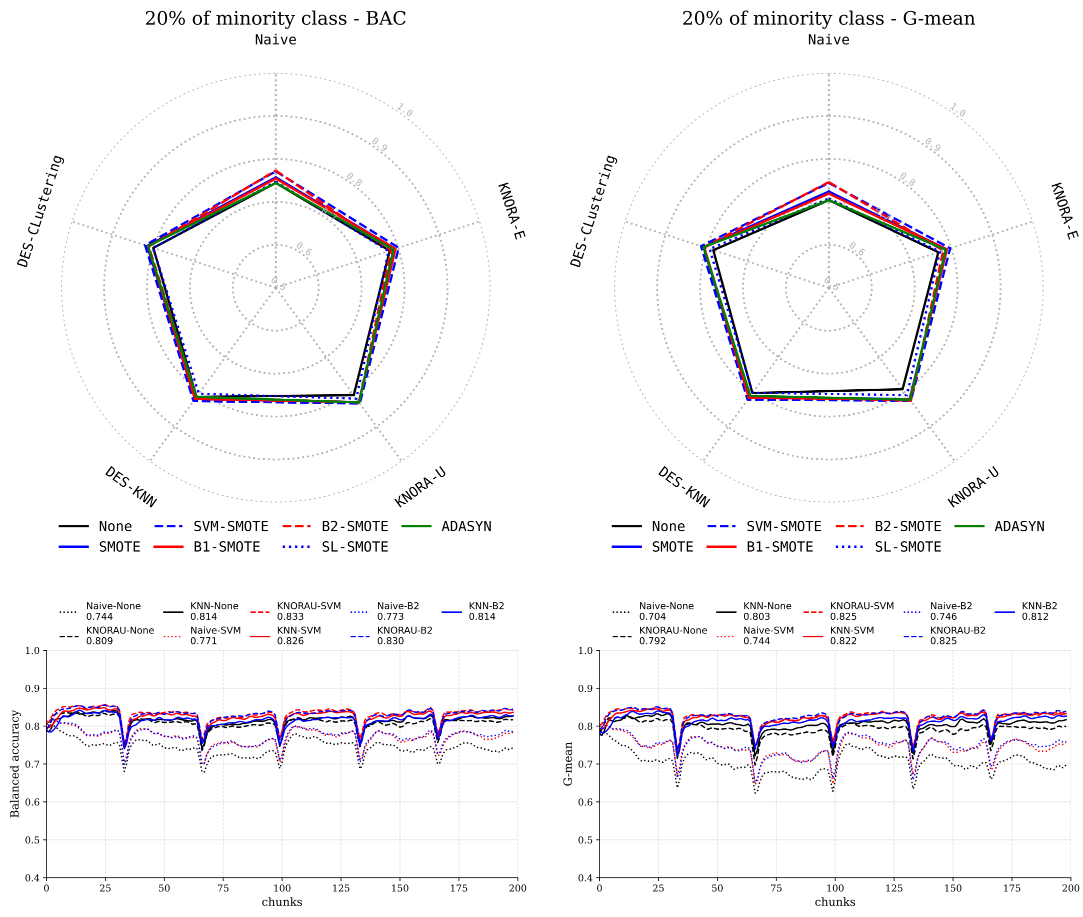


Figure 4.10: *Comparison of different sampling approaches for different classifier ensembles with respect to performance measures (*BAC *and* $Gmean_s$*) for imbalance ratio* $2:8$*.*

Based on the statistical analysis we can see that for the $1:9$ imbalance ratio, according to BAC, DES-KNN was the best performing method without the use of any preprocessing. In cases where DES was coupled with preprocessing methods, KNORA-U performed best except for the use of SL-SMOTE, where it was not statistically better than DES-KNN. According to $Gmean_s$ for $1:9$ IR DES-KNN was statistically the best dynamic ensemble selection method. For the *Borderline2*-SMOTE preprocessing method, both DES-KNN and KNORA-U performed statistically similar. The best preprocessing methods were SVM-SMOTE and *Borderline2*-SMOTE.

For the $2:8$ IR, both in terms of BAC and $Gmean_s$, KNORA-U performed best when paired with any preprocessing method. If no data preprocessing was used, DES-KNN performed statistically significantly best. As for the preprocessing methods, in most cases SVM-SMOTE was statistically significant, *Borderline2*-SMOTE performed best for *Support Accumulation* of the whole classifier pool.

For the $3:7$ imbalance ratio, KNORA-U again proved to be the statistically significantly best *Dynamic Ensemble Selection* method. The only exception (according to $Gmean_s$)

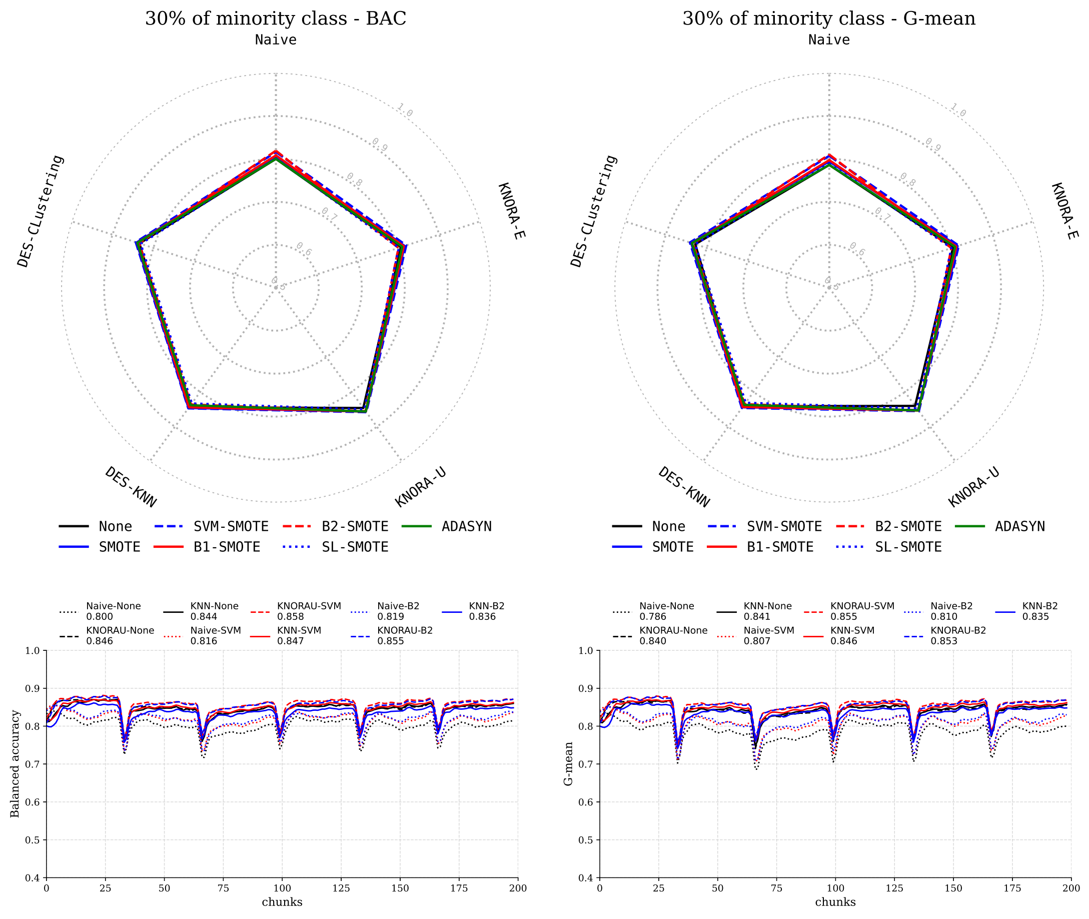


Figure 4.11: *Comparison of different sampling approaches for different classifier ensembles with respect to performance measures (*BAC *and* $Gmean_s$*) for imbalance ratio* $3:7$.

was the case where no preprocessing was used, then DES-KNN works best. By both measures, the best data preprocessing method for DES was SVM-SMOTE. *Borderline2-*SMOTE again performed the best for *support accumulation*.

In the case of $4:6$, IR was the statistically significantly best KNORA-U method in each case according to both BAC and $Gmean_s$. *Borderline2-*SMOTE worked best for *support accumulation* and in the remaining cases SVM-SMOTE was statistically the best preprocessing method.

Experiment 2 – *Concept drift* type impact

Evaluation of the DESISC in the case of different *concept drift* types (*sudden* or *incremental*) focused on the streams with high imbalance ratios (i.e., $1:9$ and $2:8$), typical for the real-life decision tasks. The comparison is shown in Figure 4.13 and 4.14. The results of statistical analysis conducted in Experiment 2 is presented in Tables 4.5 and 4.6. Bold indicates the statistically significantly best combination method, while brackets are used to denote the statistically significantly best preprocessing algorithm for a given

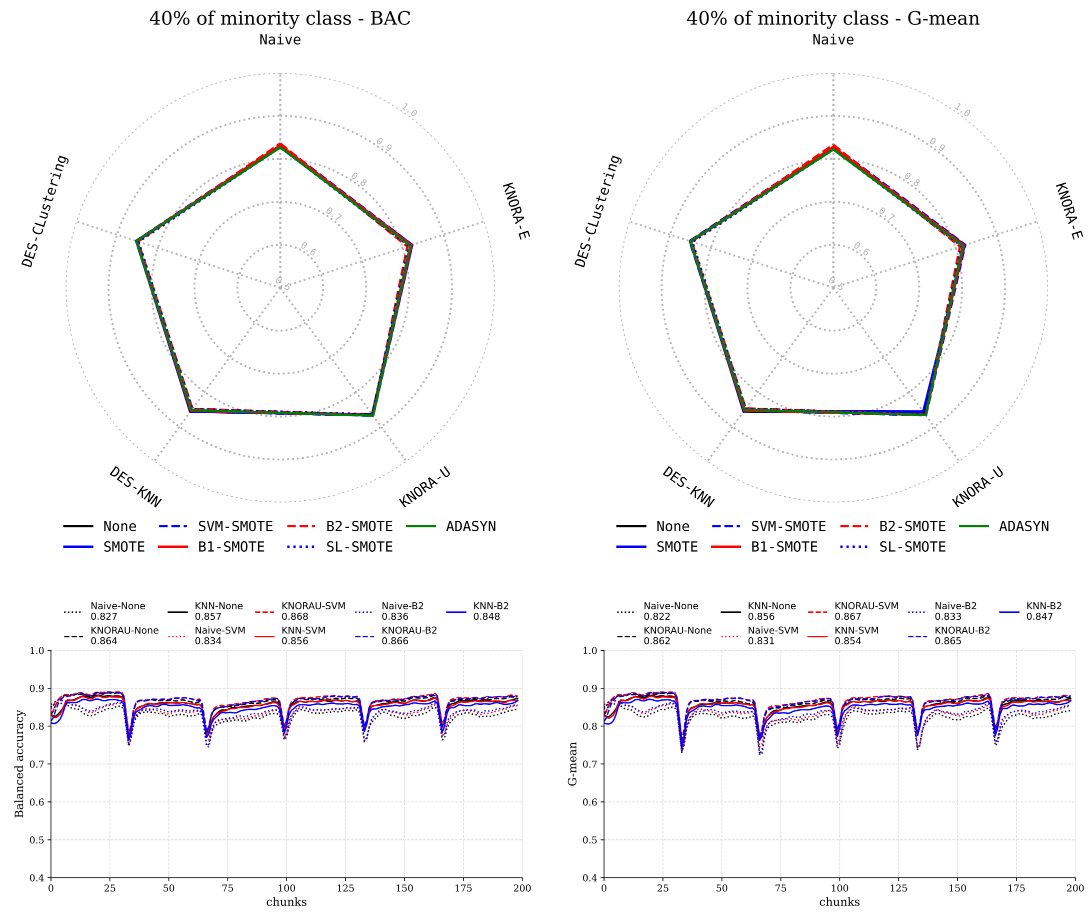


Figure 4.12: *Comparison of different sampling approaches for different classifier ensembles with respect to performance measures (*BAC *and* $Gmean_s$*) for imbalance ratio* 4 : 6.

combination strategy. Small numbers under the results indicate the indexes of methods that are statistically significantly outperformed by the considered combination strategy (best in row), while small letters stand for preprocessing methods that are statistically significantly outperformed by the considered one (best in column). Statistical analysis was conducted using the *Wilcoxon Signed Rank Test* ($p \leq .05$).

For sudden drift, in terms of both measures, DES-KNN was statistically the best without the use of any preprocessing method and KNORA-U was statistically leading when paired with every oversampling method. *Borderline2*-SMOTE was the best for *support accumulation* and for KNORA-U according to $Gmean_s$, for the rest of *Dynamic Ensemble Selection* methods SVM-SMOTE performed the best.

Finally, for incremental drift, according to BAC, DES-KNN performed statistically significantly best without the use of preprocessing and for the SL-SMOTE while KNORA-U was the best for other oversampling techniques. SVM-SMOTE was the best preprocessing method for KNORA-E, DES-KNN and DES-*Clustering* and *Borderline2*-SMOTE performed the best coupled with *support accumulation* and KNORA-U. According to $Gmean_s$,

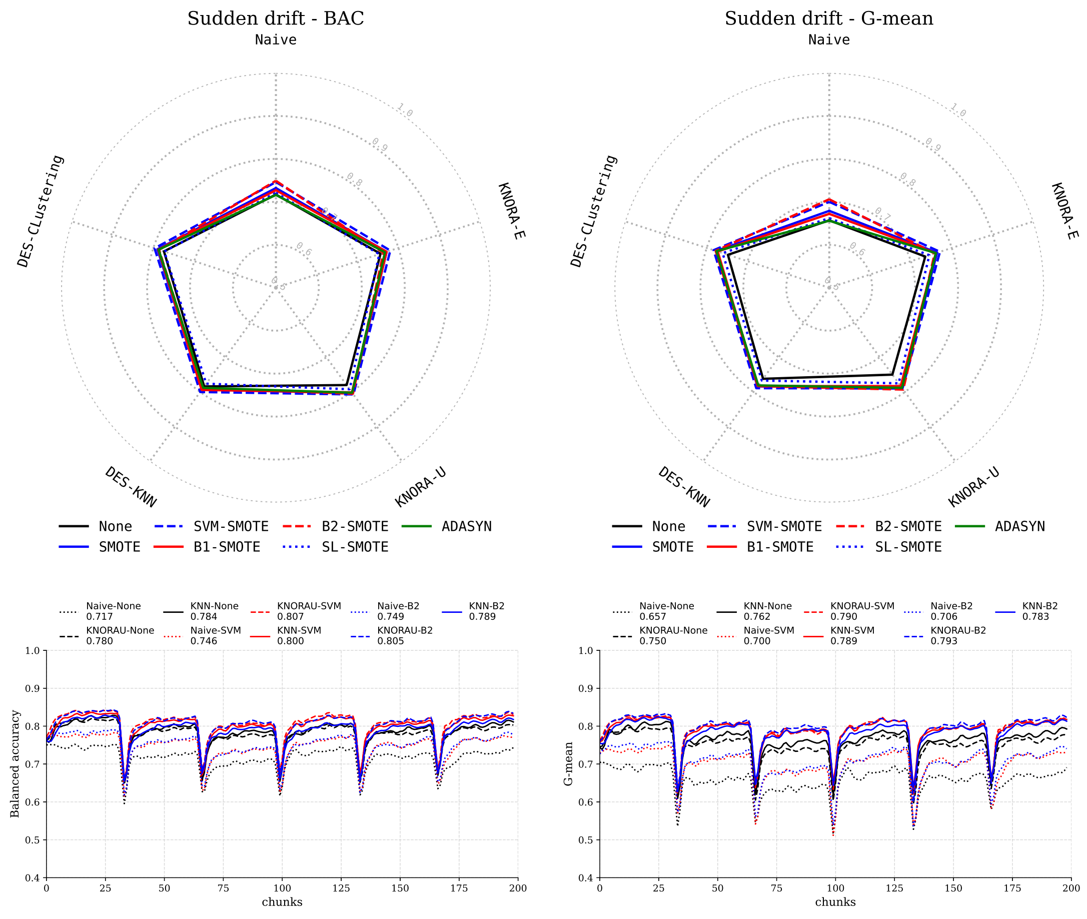


Figure 4.13: *Comparison of different sampling approaches for different classifier ensembles with respect to performance measures (*BAC *and $Gmean_s$) for sudden drift.*

KNORA-U was statistically leading DES method for *Borderline2*-SMOTE and ADASYN while DES-KNN was statistically significant for all other preprocessing techniques. SVM-SMOTE worked best with KNORA-E and DES-KNN, *Borderline2*-SMOTE proved to be statistically significant for *support accumulation*, KNORA-U and DES-*Clustering*.

Observations

n general, the order of the approaches presented in terms of performance, beginning with the worst, is as follows: (*i*) *support accumulation* without using preprocessing methods, (*ii*) *support accumulation* combined with preprocessing, (*iii*) dynamic ensemble selection methods without preprocessing, (*iv*) DES methods coupled with preprocessing methods. The lower the imbalance ratio, the smaller the differences between the approaches, but the order is maintained. The conducted experiments showed that the best performing DES method among the considered strategies across all tested imbalance ratios is the KNORA-U, which uses the weighted voting scheme. Since the KNORA-*Union* method selects all the base models that are able to correctly classify at least one instance in

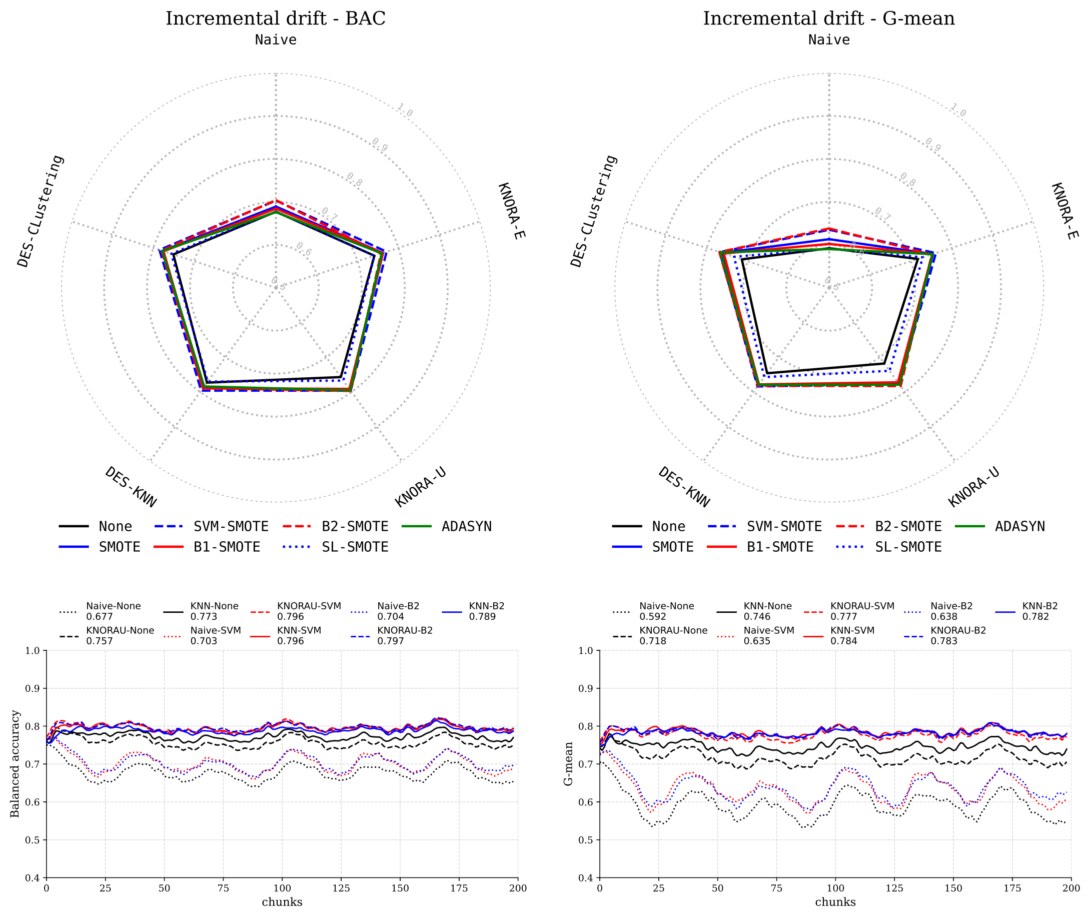


Figure 4.14: *Comparison of different sampling approaches for different classifier ensembles with respect to performance measures (*BAC *and* $Gmean_s$*) for incremental drift.*

the local competence region and then combines them based on the weighted voting, where the number of votes equals the number of correctly detected samples, it allows us to select both an accurate and a diverse ensemble. Since these two properties are the determinants of a good classifier ensemble model, they may be the reason for high results of this *Dynamic Ensemble Selection* method. Worth mentioning is also the DES-KNN, which is doing well for high imbalance ratios, especially for the 10% of minority class and for incremental drift in terms of $Gmean_s$. DES-KNN performs the best for high IR (10 and 20% of minority class) in case of not using any preprocessing method. The worst performing DES method, for low IR (30 and 40%) worse even than *support accumulation*, was KNORA-E. This may be due to the fact, that the local oracles are found only for competence regions with a significantly reduced size, which negatively affects the performance.

Based on the results achieved by DES-KNN and DES-*Clustering* methods it may suspected that the *k-Nearest Neighbors* technique is better suited for defining the local region of competence in case of imbalanced data streams than the clustering technique. Despite

Table 4.5: *Results of the Wilcoxon Signed Rank Test for various types of concept drift in relation to* BAC.

Sudden drift	SACC (1)	KNORA-E (2)	KNORA-U (3)	DES-kNN (4)	DES-Cl (5)
None ($_a$)	0.717 – *g*	0.756 1 *f*	0.780 1,2,5 –	**0.784** *All* *f*	0.774 1,2 –
SMOTE ($_b$)	0.732 – *a,d,f,g*	0.771 1 *a,e,f,g*	**0.803** *All* *a,d,f,g*	0.793 1,2,5 *a,e,f,g*	0.790 1,2 *a,d,e,f,g*
SVM-SMOTE ($_c$)	0.746 – *a,b,d,f,g*	[0.780] 1 *All*	**[0.807]** *All* *All*	[0.800] 1,2,5 *All*	[0.797] 1,2 *All*
B1-SMOTE ($_d$)	0.727 – *a,f,g*	0.771 1 *a,e,f,g*	**0.801** *All* *a,f*	0.794 1,2,5 *a,e,f,g*	0.788 1,2 *a,e,f,g*
B2-SMOTE ($_e$)	[0.749] – *All*	0.763 1 *a,f*	**0.805** *All* *a,b,d,f,g*	0.789 1,2,5 *a,f,g*	0.786 1,2 *a,f*
SL-SMOTE ($_f$)	0.721 – *a,g*	0.753 1 –	**0.792** *All* *a*	0.776 1,2,5 –	0.773 1,2 –
ADASYN ($_g$)	0.716 – –	0.767 1 *a,e,f*	**0.802** *All* *a,d,f*	0.788 1,2 *a,f*	0.787 1,2 *a,f*
Incremetal drift	**SACC (1)**	**KNORA-E (2)**	**KNORA-U (3)**	**DES-kNN (4)**	**DES-Cl (5)**
None ($_a$)	0.677 – –	0.741 1 –	0.757 1,2,5 –	**0.773** *All* *f*	0.751 1,2 –
SMOTE ($_b$)	0.689 – *a,d,f,g*	0.762 1 *a,e,f,g*	**0.793** *All* *a,d,f*	0.788 1,2,5 *a,f,g*	0.778 1,2 *a,d,f*
SVM-SMOTE ($_c$)	0.703 – *a,b,d,f,g*	[0.771] 1 *All*	**0.796** *All* *a,b,d,f,g*	[0.796] 1,2,5 *All*	[0.785] 1,2 *All*
B1-SMOTE ($_d$)	0.684 – *a,f,g*	0.762 1 *a,e,f,g*	**0.791** *All* *a,f*	0.789 1,2,5 *a,e,f,g*	0.775 1,2 *a,f*
B2-SMOTE ($_e$)	[0.704] – *All*	0.757 1 *a,f*	**[0.797]** *All* *All*	0.789 1,2,5 *a,f,g*	0.780 1,2 *a,b,d,f,g*
SL-SMOTE ($_f$)	0.677 – *g*	0.741 1 –	0.767 1,2,5 *a*	**0.770** *All* –	0.756 1,2 *a*
ADASYN ($_g$)	0.676 – –	0.759 1 *a,e,f*	**0.795** *All* *a,b,d,f*	0.784 1,2,5 *a,f*	0.778 1,2 *a,d,f*

Table 4.6: *Results of the Wilcoxon Signed Rank Test for various types of concept drift in relation to* $Gmean_s$.

Sudden drift	SACC (1)	KNORA-E (2)	KNORA-U (3)	DES-kNN (4)	DES-Cl (5)
None ($_a$)	0.657 – *g*	0.735 1 –	0.750 1,2,5 –	**0.762** *All* –	0.748 1,2 –
SMOTE ($_b$)	0.679 – *a,d,f,g*	0.764 1 *a,d,e,f,g*	**0.787** *All* *a,d,f*	0.784 1,2,5 *a,d,e,f,g*	0.777 1,2 *a,d,f*
SVM-SMOTE ($_c$)	0.700 – *a,b,d,f,g*	[0.771] 1 *All*	**0.790** *All* *a,b,d,f,g*	[0.789] 1,2,5 *All*	[0.783] 1,2 *All*
B1-SMOTE ($_d$)	0.672 – *a,f,g*	0.761 1 *a,e,f*	**0.783** *All* *a,f*	0.783 1,2,5 *a,f,g*	0.773 1,2 *a,f*
B2-SMOTE ($_e$)	[0.706] – *All*	0.758 1 *a,f*	**[0.793]** *All* *All*	0.783 1,2,5 *a,d,f,g*	0.778 1,2 *a,d,f,g*
SL-SMOTE ($_f$)	0.662 – *a,g*	0.745 1 *a*	**0.775** *All* *a*	0.767 1,2,5 *a*	0.761 1,2 *a*
ADASYN ($_g$)	0.656 – –	0.761 1 *a,e,f*	**0.788** *All* *a,b,d,f*	0.781 1,2,5 *a,f*	0.776 1,2 *a,d,f*
Incremetal drift	**SACC (1)**	**KNORA-E (2)**	**KNORA-U (3)**	**DES-kNN (4)**	**DES-Cl (5)**
None ($_a$)	0.592 – –	0.717 1 –	0.718 1,2,5 –	**0.746** *All* –	0.714 1 –
SMOTE ($_b$)	0.613 – *a,d,f,g*	0.754 1 *a,d,e,f,g*	0.775 1,2,5 *a,d,f*	**0.780** *All* *a,d,f,g*	0.763 1,2 *a,d,f*
SVM-SMOTE ($_c$)	0.635 – *a,b,d,f,g*	[0.761] 1 *All*	0.777 1,2,5 *a,b,d,f*	**[0.784]** *All* *All*	[0.768] 1,2 *a,b,d,f,g*
B1-SMOTE ($_d$)	0.602 – *a,f,g*	0.752 1 *a,e,f*	0.772 1,2,5 *a,f*	**0.778** *All* *a,f*	0.758 1,2 *a,f*
B2-SMOTE ($_e$)	[0.638] – *All*	0.751 1 *a,f*	**[0.783]** *All* *All*	0.782 1,2,5 *a,b,d,f,g*	[0.769] 1,2 *a,b,d,f,g*
SL-SMOTE ($_f$)	0.591 – –	0.729 1 *a*	0.739 1,2,5 *a*	**0.757** *All* *a*	0.733 1,2 *a*
ADASYN ($_g$)	0.590 – –	0.753 1 *a,e,f*	**0.779** *All* *a,b,c,d,f*	0.779 1,2,5 *a,f*	0.766 1,2 *a,b,d,f*

the higher computational cost, kNN allows for more precise estimation of the region of competence which leads to more possible ensemble configurations for classifying new instances.

On the other hand, SVM-SMOTE and *Borderline2*-SMOTE have proven to be the preferred preprocessing strategies for the used dynamic ensemble selection methods. The combination of KNORA-U or DES-KNN with one of those preprocessing methods always leads to the best classification performance.

Answers to research questions

The answers to the previously formulated research questions are as follows:

Q1. Can the use of Dynamic Selection, taking into account the local competencies of the base classifiers, improve the ensemble's performance in the case of the imbalanced data stream with *concept drift*?

A1. The obtained results and statistical analysis confirmed, that the use of Dynamic Selection may improve the ensemble's performance when dealing with the drifting imbalanced data stream classification task.

Q2. Can combining DES with data preprocessing improve the ensemble's performance in the case of the imbalanced data stream with *concept drift*?

A2. The conducted experiments confirmed, that combining DES with preprocessing improves DESISC performance when compared to the methods employing only one of these concepts.

Q3. Which DES methods and preprocessing techniques are best suited for the classification of a data stream with a given *concept drift* type and *Imbalance Ratio*?

A3. The results obtained showed that regardless of the *Imbalance Ratio* and the type of concept drift, the statistically significantly best performing DES method was almost always KNORA-U. The only exceptions were IR of $1 : 9$ and incremental concept drift, where in terms of $Gmean_s$, DES-KNN performed best. The best preprocessing techniques, regardless of the *Imbalance Ratio* and the type of *concept drift*, turned out to be SVM-SMOTE or B2-SMOTE.

4.3 DES and *Stratified Bagging* for Imbalanced Stream Classification

This section proposes an extension of the previously introduced DESISC framework with the generation of base classifiers using stratified bagging. This idea alludes to the article in which Roy et al. proposed a combination of DES and preprocessing for the classification of static imbalanced data [198]. Here, however, due to the promising results achieved by DSEISC, it was decided to use a *bootstrapping* approach to classify highly imbalanced data streams with *concept drift* occurrence. The motivation to use *Stratified Bagging* to generate a classifier pool was the potential possibility of obtaining a more diverse pool of base models, which may increase the chances of *Dynamic Ensemble Selection* methods to find experts in local regions of the feature space. This led to the proposition of a framework called *Dynamic Ensemble Selection for Imbalanced Stream Classification using Stratified Bagging* (DESISC-SB).

DESISC-SB framework

Here, the previously proposed DESISC framework is combined with the *Stratified Bagging*. This is to allow the generation of classifier ensemble based on each individual highly imbalanced data chunk. The use of *bootstrapping* in the process of classifier pool generation enables *Dynamic Ensemble Selection* on two levels: (i) bagging classifiers level and (ii) all base classifiers level.

Each bagging classifier Ψ_k consists of n base estimators. Let ψ_k^i denote the ith base model forming the kth bagging classifier. Each base classifier ψ_k^i is build using the $\mathcal{LS}_k^i$ learning set which is produced by preprocessing the ith stratified bootstrap $\mathcal{SB}_k^i$ from $\mathcal{DS}_k$. Details are provided in the STRATIFIEDBAGGING(DS_k) method description. DSEL$_k$ stands for the *dynamic selection dataset* for the kth data which in this case in the previously preprocessed data chunk $\mathcal{DS}_{k-1}$. One bagging classifier Ψ_k is generated based on each incoming data chunk $\mathcal{DS}_k$ and added to the bagging classifier pool Π. As the proposed framework is based on the *Streaming Ensemble Algorithm* (SEA) [221], when the maximum bagging classifiers pool size (n_{max}) is exceeded after adding a new model, the worst one, according to the *balanced accuracy* metric, will be removed from the pool. The DESISC-SB framework is presented in Figure 4.15 and the pseudoode is shown separately for the training and prediction phase in Algorithms 10 and 11.

Let us shortly describe the methods used in pseudocode:

- PREPROCESS() – applies the chosen preprocessing technique to the provided data,

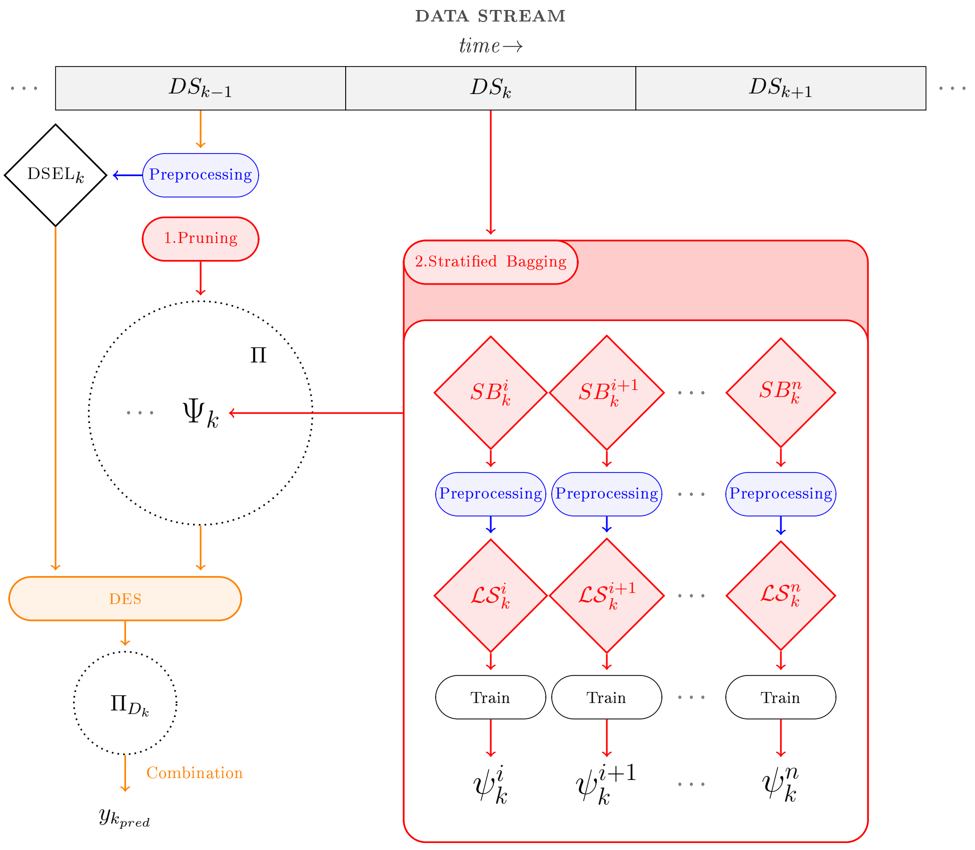


Figure 4.15: *The framework for generating the classifier pool and preparing* DSEL *for the dynamic selection process. The red arrows follow the training phase (Algorithm 10), while the orange arrows depict the prediction phase (Algorithm 11).*

- StratifiedBagging() – generates the bagging classifier for kth data chunk. Each bootstrap is generated by sampling with replacement both minority and majority classes separately in such a way that preserves the number of instances of both classes in the original data chunk. The final decision of Ψ_k is made based on the aggregation of the support functions of n individual classifiers according to the sum rule [76]. When coupled with preprocessing techniques, preprocess() method is called on each bootstrap according to Figure 4.15,
- predict() – uses a given classifier pool (or list of ensembles in case of dynamic selection) to classify each instance in given data chunk,
- DynamicSelection() – uses a given dynamic ensemble selection method to generate a list of ensembles for classifying each test instance. In this work DES can be performed on two levels - bagging classifier level or base estimators level,
- pruneWorst() – removes the worst-performing base classifier from the pool if the maximum bagging classifier pool size (n_{max}) is exceeded after adding a new model.

Algorithm 10 Training phase of the DESISC-SB framework

```
Input:
    $Stream$ – data stream,
    $n_{max}$ – maximum fixed size of the bagging classifier pool,
Symbols:
    $\mathcal{S}_k$ – set of evaluation metric values for each base classifier,
    $\mathcal{DS}_k$ – data chunk,
    $\Psi_k$ – bagging classifier,
    $\Pi$ – bagging classifiers pool.

1: $\Pi \leftarrow \varnothing$
2: for each $k, \mathcal{DS}_k = \{x_k^1, x_k^2, \ldots, x_k^N\}$ in $Stream$ do
3:     if $k <= 1$ then                                    ▷ First data chunk.
4:         $\Psi_k \leftarrow$ STRATIFIEDBAGGING($\mathcal{DS}_k$)          ▷ Bagging classifier generation
5:         $\Pi \leftarrow \Psi_k$                          ▷ Adding bagging classifier to the pool
6:     else                                     ▷ Third and all subsequent data chunks.
7:         $\mathcal{S}_k$ = EVALUATE($\Pi, \mathcal{DS}_k$)
8:         if $\mid \Pi \mid > n_{max} - 1$ then     ▷ Removing worst classifier if $n_{max}$ is exceeded.
9:             $\Pi \leftarrow$ PRUNEWORST($\Pi, \mathcal{S}_k$)
10:        $\Psi_k \leftarrow$ STRATIFIEDBAGGING($\mathcal{DS}_k$)
11:        $\Pi \leftarrow \Psi_k$
12: end for
```

Algorithm 11 Prediction phase of the DESISC-SB frameworks

```
Input:
    $Stream$ – data stream,
    $\Pi$ – pool of bagging classifiers.
Symbols:
    $DS_k$ – data chunk,
    $\Pi_{D_k}$ – classifier ensemble selected using dynamic selection,
    $\mathcal{DSEL}_k$ – dynamic ensemble selection dataset for the $k$th data chunk.

1: for each $k, \mathcal{DS}_k = \{x_k^1, x_k^2, \ldots, x_k^N\}$ in $Stream$ do
2:     if $k == 0$ then                                      ▷ First data chunk
3:         $Pass$                                              ▷ No prediction
4:     else if $k == 1$ then                                ▷ Second data chunk
5:         $Decision \leftarrow$ PREDICT($\mathcal{DS}_k, \Pi$)          ▷ Prediction using the whole pool
6:         $\mathcal{DSEL}_{k+1} \leftarrow$ PREPROCESS($\mathcal{DS}_k$)          ▷ Storing $\mathcal{DSEL}$ for next step
7:     else                                     ▷ Third and all subsequent data chunks
8:         $\Pi_{D_k} \leftarrow$ DYNAMICSELECTION($\Pi, \mathcal{DSEL}_k, \mathcal{DS}_k$)       ▷ Dynamic selection
9:         $Decision \leftarrow$ PREDICT($\mathcal{DS}_k, \Pi_{D_k}$)        ▷ Prediction using selected pool
10:        $\mathcal{DSEL}_{k+1} \leftarrow$ PREPROCESS($\mathcal{DS}_k$)
11: end for
```

The step by step description is as follows. At the start. the classifier pool Π is empty and the first bagging classifier (Ψ_0) is generated using STRATIFIEDBAGGING() method on the first data chunk (Algorithm 10 steps 4 and 5). When the second chunk arrives, it is classified using the PREDICT() function (Algorithm 11 step 5) and then used to generate a new bagging model Ψ_k, which is added to the ensemble (Algorithm 10 steps 4 and 5). $\mathcal{DS}_1$ is preprocessed using PREPROCESS() method and stored as the DSEL for the dynamic selection process in the future (Algorithm 11 step 6). Then, with the arrival of each new data chunk, following steps are performed:

- In Algorithm 11 step 8, previously stored DSEL is used in the dynamic selection process for each instance in $\mathcal{DS}_k$ (DynamicSelection() method),
- In Algorithm 11 step 9, the list of ensembles selected by DES method is used to classify all the instances in the current data chunk,
- In Algorithm 10 steps 8 and 9, the PruneWorstClassifier() method is used to prune the classifier pool if the fixed size n_{max} is exceeded,
- In Algorithm 10 steps 10 and 11 a new bagging classifier Ψ_k is generated using kth data chunk and added to the pool Π,
- Finally, in Algorithm 11 step 10, the current data chunk $\mathcal{DS}_k$ is preprocessed and stored in order to use it as DSEL in the next iteration.

Computational and memory complexity analysis

The computationl complexity of DESISC-SB framework is largely adequate to the complexity of DESISC, as it is also based on the methods of *Dynamic Ensemble Selection* as well as on preprocessing techniques (both *oversampling* and *undersampling*). The key factors affecting the computational complexity of the presented approaches are, respectively, the number of models in the classifier pool for *Dynamic Selection* algorithms and the number of problem instances in a single data chunk in the case of preprocessing techniques.

Based on preliminary observations, it was established that the DES methods (both KNORA-U and KNORA-E) have a linear time complexity of $O(n)$ depending on the number of base classifiers in the pool. The preprocessing techniques used in the work have, respectively, the logarithmic complexity of $O(log\ n)$ (ROS and RUS), the quadratic complexity of $O(n^2)$ (*Borderline2*-SMOTE) [260], and the complexity of $O(n\ log\ n)$ (CNN).

Stratified Bagging performs sampling with replacement for each class with computational complexity of $O(\mid i \mid n)$, where $\mid i \mid$ is the cardinality of the ith class and n denotes the number of *bootstraps* (number of base models in bagging classifier) [79].

Because a fixed size of the data chunk N is always set, the complexity of the proposed algorithms depends only on the number of classifiers from which the selection is made (denoted as $\mid \Pi \mid$). Methods that perform dynamic selection at the level of base classifiers have a linearly higher computational complexity than those that do it at the level of bagging classifiers.

4.3.1 Experimental evaluation

Here, the experimental set-up plan for the DSEISC-SB-SB framework will be presented along with the motivation, objectives of the individual experiments, and the results obtained.

Research questions

The experiments were designed to answer the following questions:

Q1. Which *Dynamic Ensemble Selection* methods perform best while dealing with the *concept drift* occurrence?

Q2. Does performing *Dynamic Ensemble Selection* at the level of all generated base models (including those forming individual bagging classifiers) allow *dseisc-sb* to achieve better performance when compared to *Dynamic Selection* performed only at the level of bagging classifiers?

Q3. Can methods combining data preprocessing and *Dynamic Ensemble Selection* outperform *state-of-the-art* batch-based and online classifiers for difficult data stream classification task?

Goals of the experiments

Experiment 1 – Dynamic selection level

The main purpose of the first experiment is, due to a large number of methods, the pre-selection of further used dynamic ensemble selection approaches. Dynamic selection without the use of preprocessing techniques is evaluated for the potential to classify highly imbalanced data. Based on the results obtained from this shortened experiment in which the results are presented only for the highest tested *Imbalance Ratio* and *stream-learn* generated data streams, a pool of classifiers will be selected, on which DES methods will be used later for a given type of base classifier (i.e., bagging level or the level of all base classifiers present in the pool).

Experiment 2 – Pairing DES *with preprocessing techniques*

The second experiment aims to examine how two previously chosen DES methods perform based on the preprocessing technique with which they were paired compared to using solely dynamic selection. We divided the experiment into two parts, i.e., oversampling and undersampling. After analyzing the results obtained, one preprocessing method will be selected from both groups, which then will be used in subsequent experiments. Again, this is a shortened experiment in which the results only for 3% of the minority class and stream generated using the *stream-learn* package are presented.

Additionally, the behavior of each approach during the *sudden concept drift* occurrence was analyzed. using the *restoration time* and *maximum performance loss* metrics.

Experiment 3 – Comparison with state-of-the-art
In the third experiment, two previously selected dynamic selection methods and two preprocessing techniques are compared with *state-of-the-art* online data stream classification approaches based on the notion of offline *Bagging*, as well as with the chunk-based stream classification methods. Because online methods require a base classifier capable of incremental learning, a comparison was possible only for *Gaussian Naïve Bayes* and *Hoeffding Tree classifiers.*

In the case of this experiment, artificially generated data streams from both *stream-learn* and MOA were used and full results for three imbalance ratios and three types of *concept drift* are presented. Results for 10 and 20% of the minority class can be found on *GitHub*. Due to the high computational complexity of *Hoeffding Trees*, they were tested only for real data streams.

Experimental set-up
To evaluate the proposed framework 90 artificially data streams were generated with various characteristics using *stream-learn* Python library [141]. Each data stream is composed of fifty thousand instances (200 chunks, 250 instances each) described by 8 informative features, and contains a single concept drift (in the 100th data chunk). The variety of streams was ensured by generating two streams, based on the determined seeds, for each combination of the following parameters:

- *the imbalance ratio* — successively 3, 5, 10, 15 and 20% of the minority class.
- *the level of label noise* — successively 1, 3 and 5%.
- *the type of concept drift* — *sudden*, *gradual*, or *incremental.*

The remaining 45 data streams were generated using the MOA data stream mining framework [19]. While retaining the parameters mentioned above, these streams differ in the generator used and the number of attributes:

- *the generator used* — *Agrawal* (*sudden* and *gradual concept drift*) and *Hyperplane* (*incremental concept drift*).
- *the number of attributes* — 9 for the *Agrawal* generator and 10 for *Hyperplane* generator.

Additionally, this paper presents the results of experiments carried out on real data streams presented in Table 4.7.

Table 4.7: *Real data streams characteristics.*

Data stream	#Samples	#Features	IR
covtypeNorm-1-2vsAll	266 000	54	4
poker-lsn-1-2vsAll	360 000	10	10
INSECTS-abrupt_imbalanced_norm	300 000	33	19
INSECTS-gradual_imbalanced_norm	100 000	33	19
INSECTS-incremental_imbalanced_norm	380 000	33	19

Evaluation of the proposed framework was based on six metrics dedicated for imbalanced data classification problems. The experimental protocol *Test-Then-Train* [135] was used, i.e., the classification model is trained on a current data chunk and it is evaluated based on the following one. As the base estimators, four different classification models according to the *scikit-learn* implementation [187] were used. In the research on ensemble methods, large pools of classifiers, such as 100 [60] or even 1000 [204] base models, are usually considered. However, the interesting experiments regarding the prediction of the best classifier pool size for *Dynamic Selection* methods suggested that pools containing an average of 20 classifiers might perform best [196, 197]. Therefore, in order to improve the performance of DSEISC-SB and to reduce its computational complexity, the maximum size of the bagging classifier pool was set to $n_{max} = 5$ and each bagging classifier consisted of $n = 10$ base models. Batch-based reference methods use 5 bagging classifiers, each of which consists of 10 base models, while online reference methods maintain ensembles consisting of 20 base classifiers. Experiments were implemented in Python programming language and may be repeated according to source code published on *GitHub*[3].

- Evaluation metrics – *Balanced Accuracy Score* (BAC), *Gmean$_s$*, *F_1 score*, *precision*, *recall*, and *specificity*,
- Classification algorithms – *Gaussian Naïve Bayes* (GNB), *Hoeffding Tree* (HT), *k-Nearest Neighbors* classifier (*k*NN) and *Support Vector Machine* (SVM),

[3] `https://github.com/w4k2/if-des-imb-stream`

- Reference methods
  - *Online Bagging* (OB) [184], which updates each base classifier in the pool with the appearance of a new instance using the *Poisson*($\lambda = 1$) distribution.
  - *Oversampling-Based Online Bagging* (OOB) and *Undersampling-Based Online Bagging* (UOB) [243], which integrate resampling into the *Online Bagging* algorithm. This was achieved by making the λ value dependent on the proportion between classes.
  - *Learn++*.NIE (*Nonstationary and Imbalanced Environments*) and *Learn ++*.CDS (*Concept Drift with* SMOTE) [72], which extend the *Learn++*.NSE (*Non-Stationary Environments*) algorithm.
  - *Recursive Ensemble Approach* (REA) [53], which incorporates part of previous minority class samples into the current data chunk and combines base models in a dynamically weighted manner.
  - *Over/UnderSampling Ensemble* (OUSE) [91], which uses minority class instances from all previously seen data chunks and a subset of majority class present in the most recent chunk to generate new ensemble.
  - KMC [246], an ensemble-based approach, which performs, on each arriving data chunk, *undersampling* based on the *k-Means* clustering algorithm.

In total, based on the proposed framework, fifteen methods for the classification of imbalanced data streams have been distinguished in this paper. These methods differ in the applied preprocessing techniques and the dynamic selection methods used. We chose two dynamic ensemble selection methods and two preprocessing techniques for experiments:

- Dynamic ensemble selection methods – KNORA-E and KNORA-U were selected due to the relatively low complexity compared to e.g. DES-KNN, which may not be suited for data stream environment due to costly ensemble diversity calculation.
- Preprocessing techniques
  - Oversampling – *Random Oversampling* and *Borderline2*-SMOTE selected, based on experiments carried out for DESISC, as the best performing among several SMOTE variants when paired with DES for imbalanced data stream classification.
  - Undersampling – *Random Undersampling* and *Condensed Nearest Neighbour*.

In addition, the cases of no preprocessing applied and classic support accumulation of the classifier pool instead of *Dynamic Selection* are considered. The *Dynamic Ensemble*

Selection is performed in two variants – on the bagging classifiers level or the level of all base models (including those making up each bagging classifier). The variant of *Dynamic Selection* is denoted by the number after the name of DES method, 1 being bagging classifiers and 2 being all base estimators. The neighborhood size for DES methods is $k = 7$, as it is the most commonly suggested value for the local region of competence [59].

Also, to reduce the amount of information, only the most interesting results are presented, and to facilitate concluding, results for two of the four base classifiers are omitted, namely GNB and KNN, in Experiments 1 and 2. GNB achieved results remarkably close to HT, and KNN showed behavior quite similar to GNB and HT.

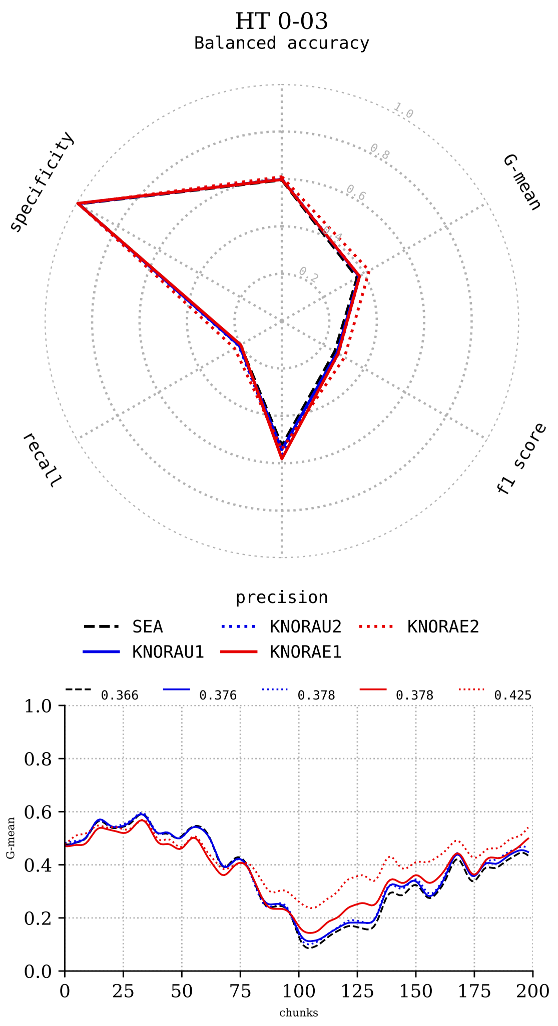
Some of the observations regarding the results obtained by the omitted models are presented in the *Observations* subsection. Runs smoothed using Gaussian filter ($\sigma = 3$) are presented for the $Gmean_s$, as it best reflects the relationships between the methods' performance.

Experiment 1 – Dynamic selection level

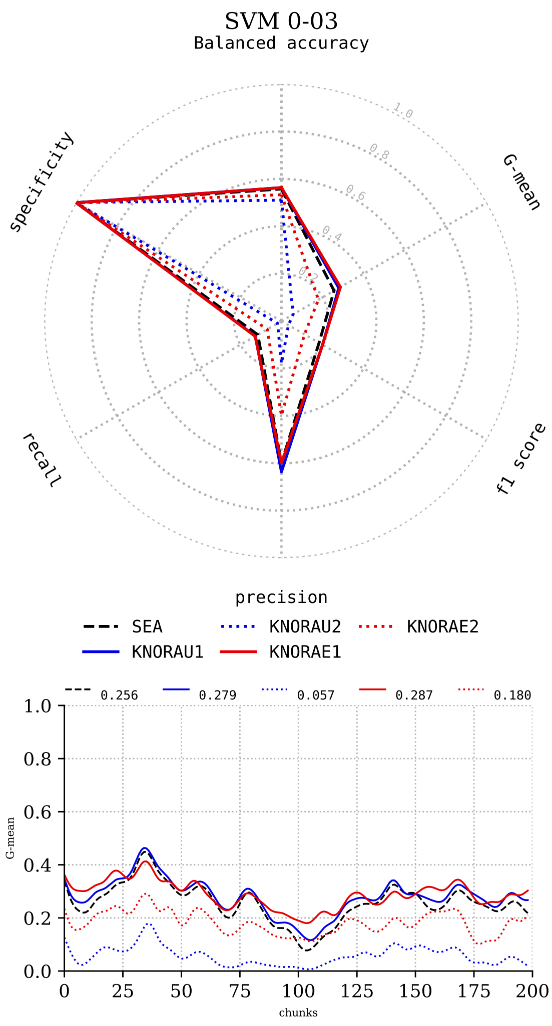
Figure 4.16 shows the results for the use of selected *Dynamic Ensemble Selection* methods at bagging classifiers level and base classifiers level, when *Hoeffding Tree* (Figure 4.16a) and *Support Vector Machine* (Figure 4.16b) were used as base models. In case of HT, radar diagrams show slight differences in terms of each metric when compared to the basic SEA as data streams with a high *Imbalance Ratio* are analyzed without using any preprocessing techniques. Despite this, KNORAE2 has an advantage in terms of $Gmean_s$, F_1 *score* and BAC.

More significant differences are visible in the presented runs, in which significantly better response to the concept drift when the KNORAE method is used (both at the level of bagging and base models) can be observed. This may be because this algorithm can select base classifiers that are local oracles in a given fragment of the feature space, which in the event of a concept change allows us to keep only the models already trained on the given concept.

In the case of the SVM classifier (Figure 4.16b), the use of DES at the base estimators level leads to a significant deterioration of the results obtained in terms of each metric except for *specificity*. This may be due to a large number of poorly differentiated classifiers in the pool. The selection methods used at the bagging classifiers level, especially KNORAU1, perform similar to SEA.



(a) HT



(b) SVM

Figure 4.16: *Experiment 1 results for* Hoeffding Tree *and* Support Vector Machine *classifiers.*

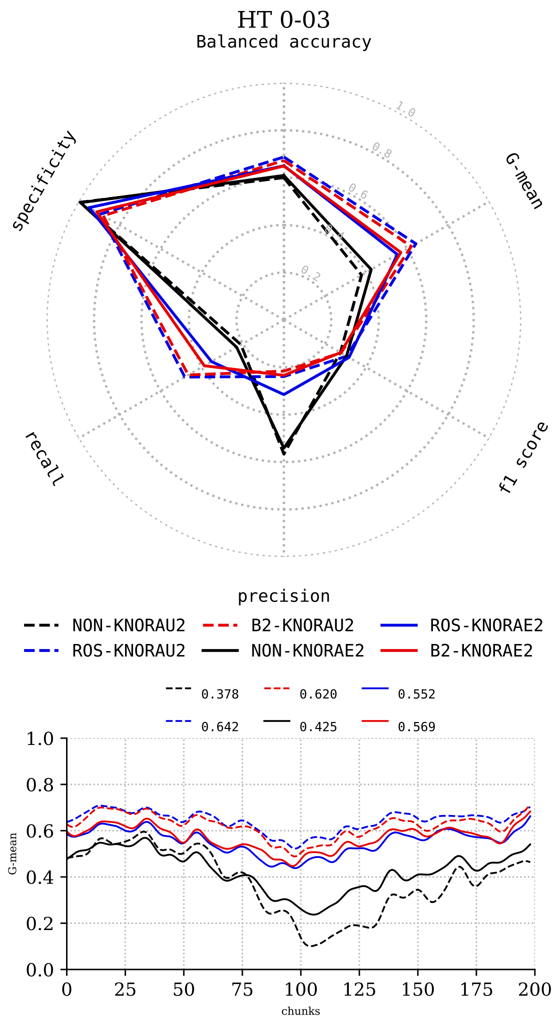
Based on the results obtained, the following methods of *Dynamic Ensemble Selection* were selected for further experiments:

- HT - KNORAU and KNORAE on the base classifiers level (KNORAU2, KNORAE2).
- SMV -KNORAU and KNORAE on the bagging classifiers level (KNORAU1, KNORAE1).

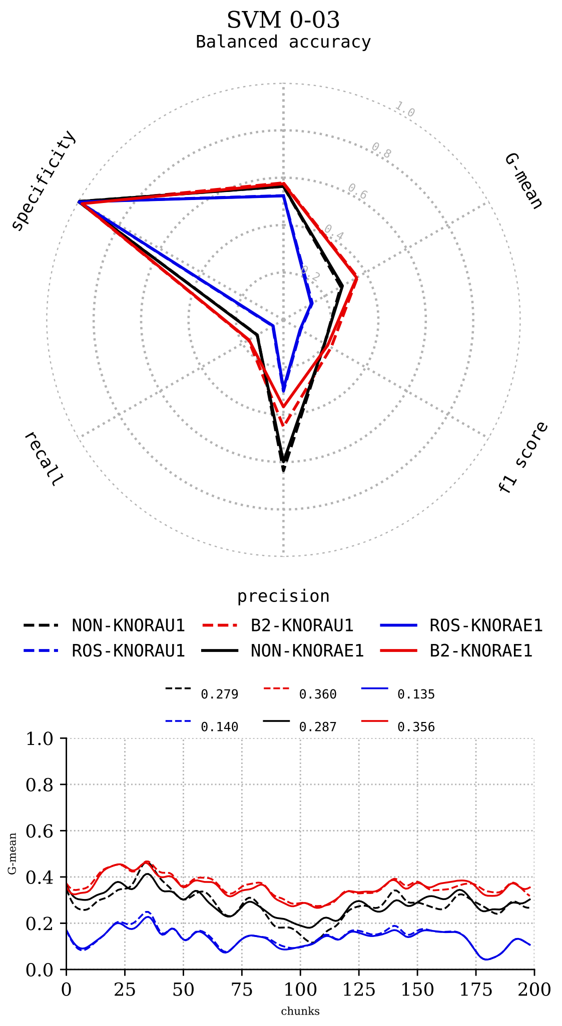
Experiment 2 – Pairing DES with preprocessing techniques

The following are the results of combining selected methods of *Dynamic Ensemble Selection* with preprocessing techniques. The experiment was divided into parts related to *oversampling* and *undersampling*.

Oversampling



(a) HT



(b) SVM

Figure 4.17: *Experiment 2.1 results for* Hoeffding Tree *and* Support Vector Machine *classifiers.*

Table 4.8: *$Gmean_s$-based performance metrics regarding sudden drift for Experiment 2.1.*

Performance metric	NON-U	ROS-U	B2-U	NON-E	ROS-E	B2-E
			HT			
performance loss	0.851	0.481	0.473	0.720	0.571	0.516
restoration time	0.023	0.012	0.013	0.017	0.010	0.009
			SVM			
performance loss	0.667	0.167	1.000	0.833	0.167	1.000
restoration time	0.012	0.008	0.010	0.008	0.007	0.010

Figure 4.17 shows the results of the combination of DES and preprocessing techniques in cases where HT or SVM was used as the base classifier. For HT the use of preprocessing leads to an increase in the *recall* at the expense of *precision* and an increase in *balanced accuracy* and $Gmean_s$. On the presented runs, it can be seen that the use of preprocessing

in conjunction with DES allows for much smaller losses in $Gmean_s$ at the time of the concept drift. This is particularly visible in the case of the *Random Oversampling* coupled with KNORAU2. Here, ROS proved to be a better oversampling method.

When SVM was employed as the base classifier (Figure 4.17b), the use of ROS caused the deterioration of all metrics except *specificity*, because duplicate instances cause a stronger shift in the decision boundary. The use of B2-SMOTE leads to a significant reduction in *precision* and a slight decrease in the F_1 *score*, while the other metrics are comparable to pure *Dynamic Selection*.

Table 4.8 contains *performance loss* and *restoration time* values in terms of $Gmean_s$ averaged over all runs, referring to *sudden* concept drift. In the case of HT, methods paired with Borderline2-SMOTE generally achieve the smallest *performance loss* and *restoration time* values. This may be due to the generation of artificial minority samples near the decision boundary. In the case of SVM classier, DES (according to the presented metrics) performs best when combined with ROS.

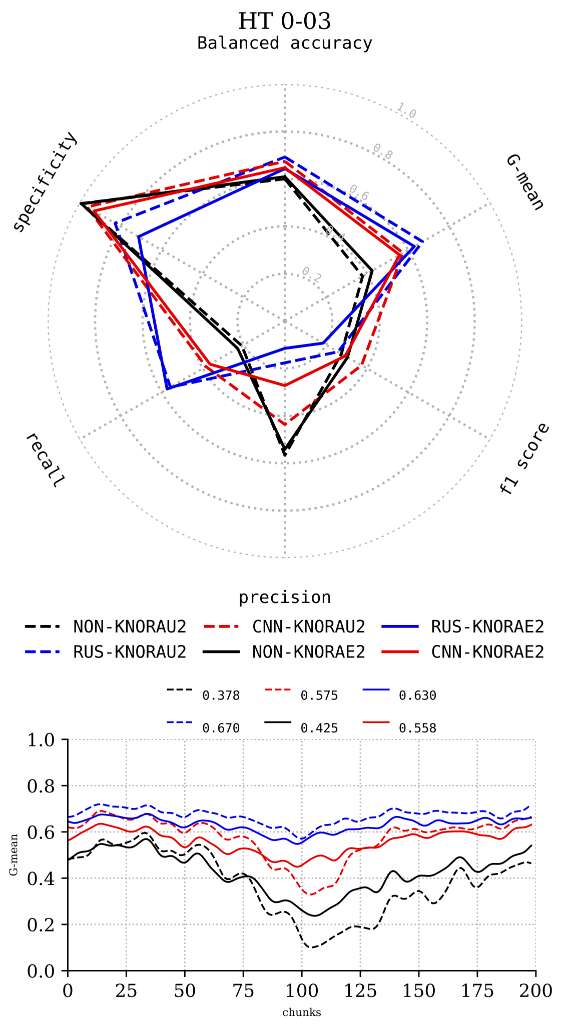
It should be noted that better performance in terms of *performance loss* and *restoration time* does not necessarily mean better classification performance. This can be observed especially in the case of SVM.

Undersampling

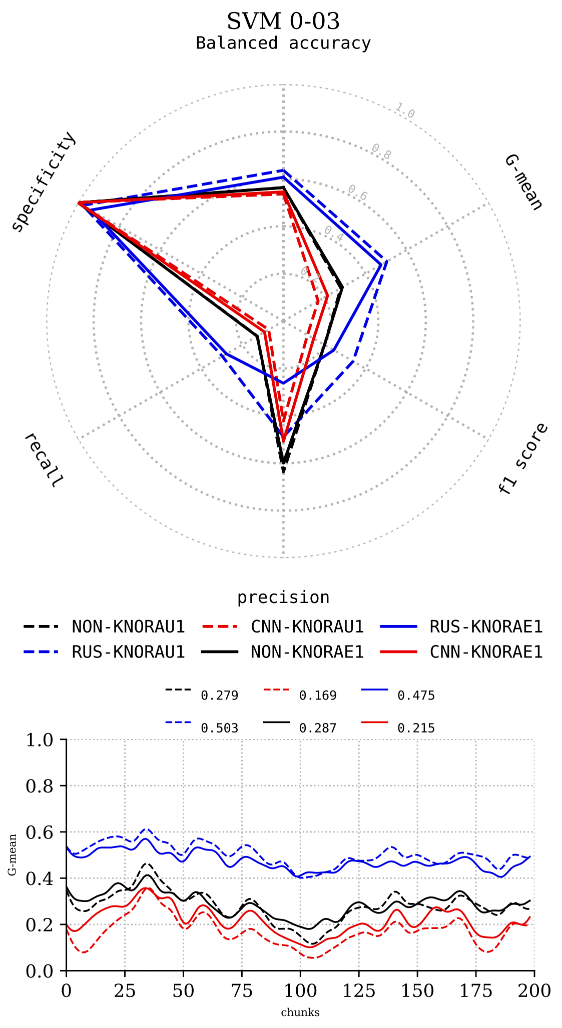
Table 4.9: *$Gmean_s$-based performance metrics regarding sudden drift for Experiment 2.2.*

Performance metric	NON-U	RUS-U	CNN-U	NON-E	RUS-E	CNN-E
				HT		
performance loss	0.851	0.578	0.663	0.720	0.466	0.598
restoration time	0.023	0.017	0.021	0.017	0.008	0.009
				SVM		
performance loss	0.667	0.833	0.500	0.833	0.924	0.500
restoration time	0.012	0.012	0.010	0.008	0.010	0.008

Figure 4.18 shows the results regarding the use of *undersampling* in combination with *Dynamic Ensemble Selection* for HT and SVM base classifiers. As can be seen, for *Hoeffding Trees*, the use of both *Random Undersampling* and *Condensed Nearest Neighbor* leads to a noticeable improvement in *recall*, *balanced accuracy* and $Gmean_s$, while reducing *precision*. In addition, RUS also leads to deterioration of F_1 *score* and *specificity*. As in the case of *oversampling* techniques, the profit from *undersampling* is best seen at the moment of the concept drift occurrence, where only a slight decrease in $Gmean_s$ can be observed. Despite the advantage of RUS in terms of this metric, a better *undersampling*



(a) HT



(b) SVM

Figure 4.18: *Experiment 2.2 results for* Hoeffding Tree *and* Support Vector Machine *classifiers.*

method for HT classifier was CNN, as it led to balanced results in terms of each of the evaluation metrics.

For SVM (Figure 4.18 b), employing RUS leads to better results, while the use of CNN practically does not cause difference when compared to the methods without preprocessing. This is due to the internal design of this *undersampling* method, which does not change the decision boundary.

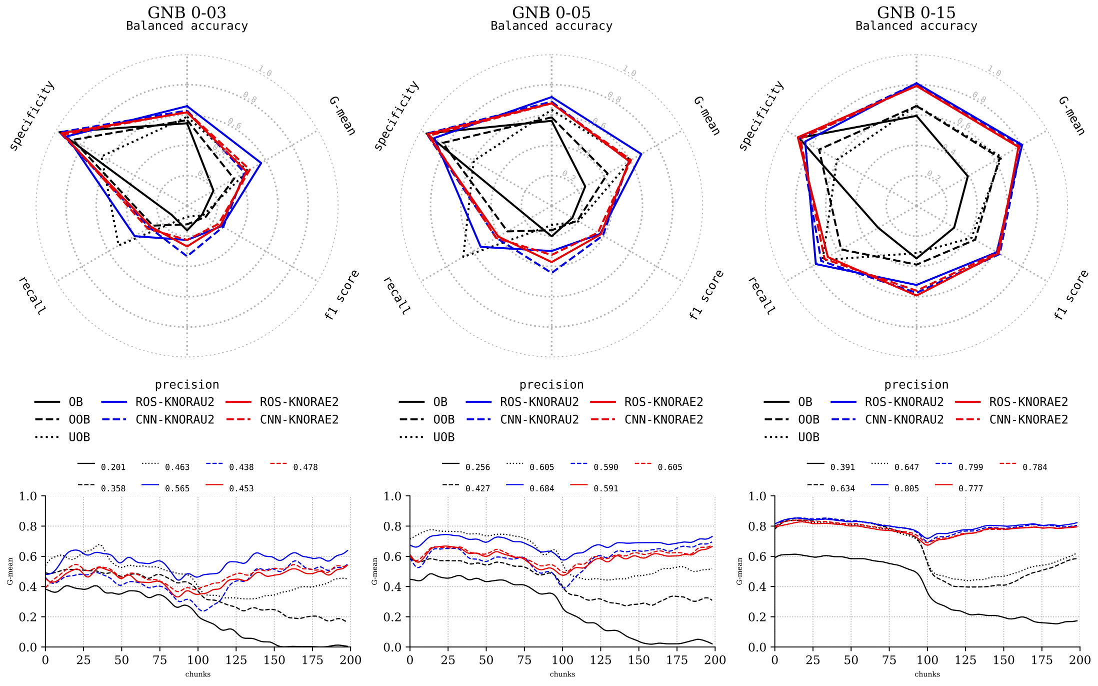
Table 4.9 presents the *performance loss* and *restoration time* values for undersampling methods. In the case of HT, RUS achieves the best values of these metrics. In the case of SVM classifier, CNN allowed DES techniques to achieve the lowest performance loss and restoration time, but simultaneously, it led to the worst classification performance.

Based on the results obtained for the HT classifiers, *Random Oversampling* and *Condensed Nearest Neighbor* were selected as the preprocessing methods for Experiment 3.

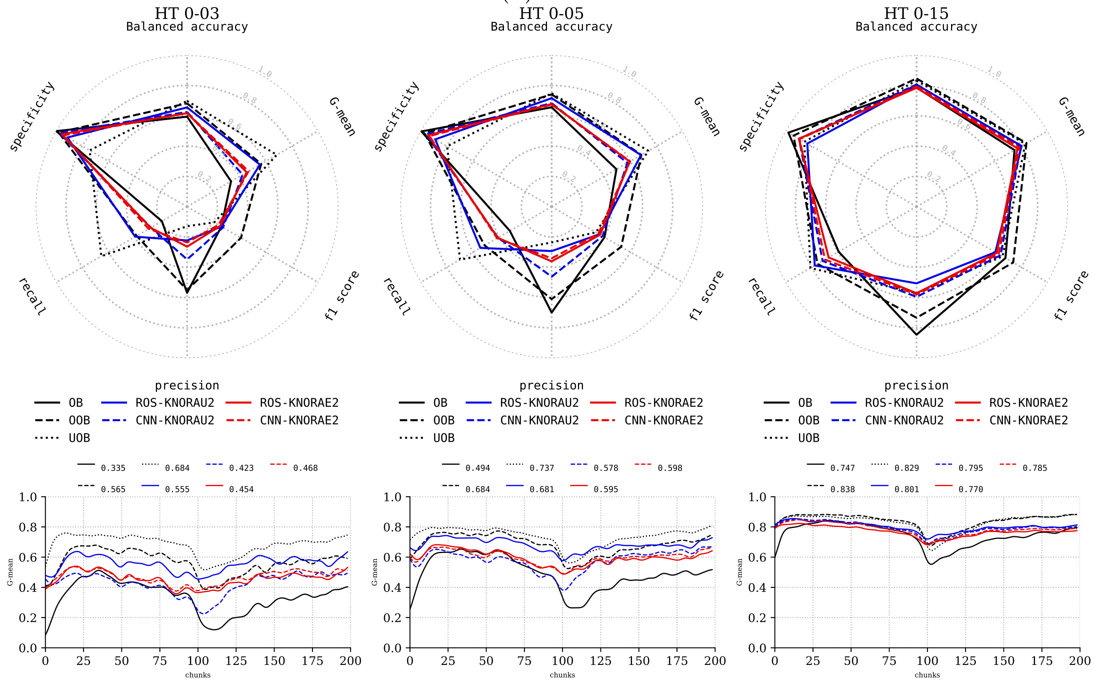
Same for *Gaussian Naïve Bayes*, for which the results were omitted due to the high similarity to the *Hoeffding Tree.*

Experiment 3 – Comparison with state-of-the-art

Online reference methods

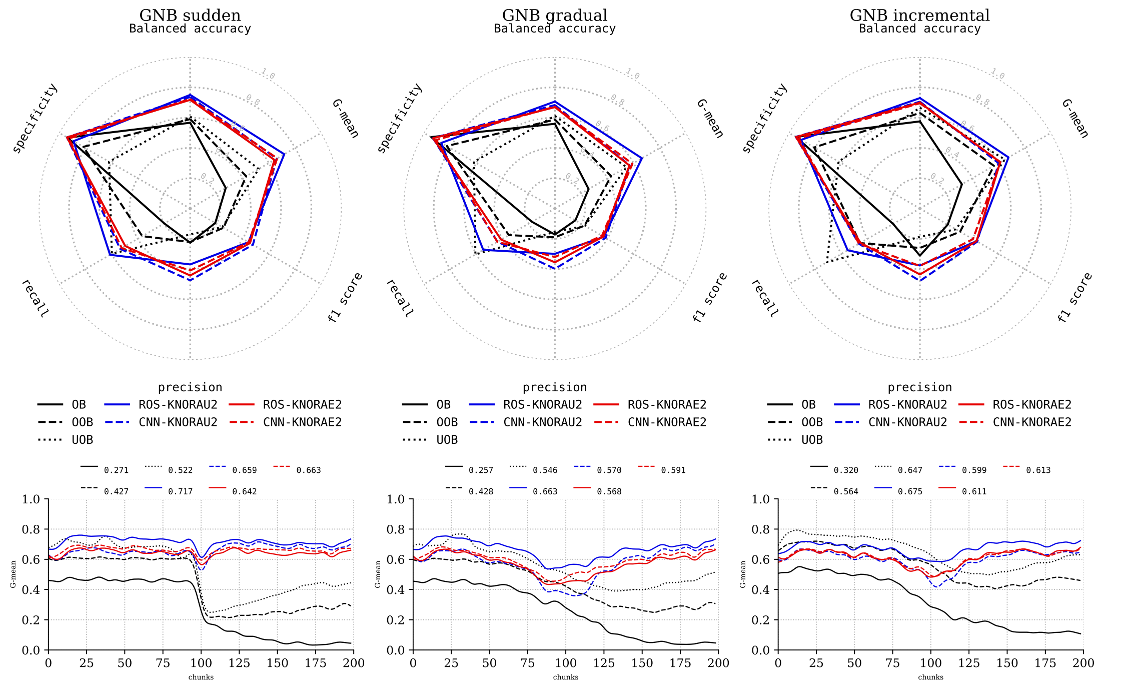


(a) GNB

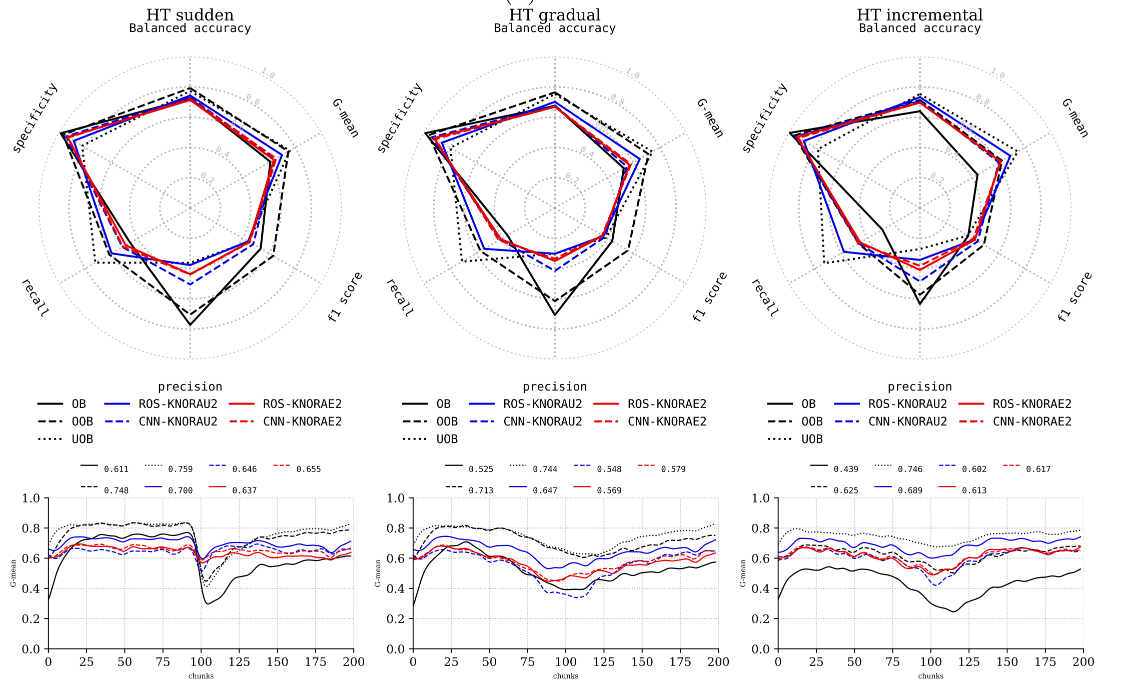


(b) HT

Figure 4.19: *Results of the experiment regarding online reference methods for various imbalance ratios.*



(a) GNB



(b) HT

Figure 4.20: *Results of the experiment regarding online reference methods for various concept drift types.*

Figure 4.19 shows a comparison of a combination of previously selected dynamic selection methods and preprocessing techniques with *state-of-the-art* online *bagging*-based methods. As these methods need base models capable of updating incrementally, this experiment was performed only for *Gaussian Naïve Bayes* and *Hoeffding Tree.*

Gaussian Naïve Bayes (Figure 4.19 a) is not suitable for online methods in the case of concept drift as $Gmean_s$ significantly decreases, because the classifier still remembers the old concept. *Online Bagging* is not able to rebuild after drift occurrence, while OOB and UOB note a lower decline and are slowly recovers thanks to built-in *oversampling* and *undersampling* methods. When it comes to the combination of DES and preprocessing, the relationships between the methods persist, but decrease with the *Imbalance Ratio*. UOB rises faster than OOB at 3 and 5% of the minority class, but when the *Imbalance Ratio* is lower both methods converge.

In the case of HT (Figure 4.19 b), it can be seen that the use of trees in online methods leads to a much smaller decrease in the $Gmean_s$ value at the moment of concept drift and leads to faster recovery. This is due to the construction of the *Hoeffding Tree* recognition model. OOB and UOB achieve better results than methods combining DES and preprocessing in terms of *balanced accuracy*, $Gmean_s$ and *recall*. In addition, UOB leads when it comes to F_1 *score* and *precision*, OOB achieves worse F_1 *score* and *specificity*. As in the case of GNB, when the *Imbalance Ratio* decreases, the results achieved by individual methods begin to converge. The OB improvement is particularly noticeable.

Figure 4.20 shows a comparison of selected batch methods with online methods in terms of concept drift type. It can be seen that the relationships shown in Figure 4.19 are also true in this case. It is noteworthy that although in the case of using *Hoeffding Tree* as the base classifier OOB and UOB perform comparably or better than the proposed batch methods, they note a more significant decrease in $Gmean_s$ and a slower recovery after *sudden concept drift*. Therefore, it can by assumed that in the case of a large number of sudden drifts occurring in the data stream, the use of batch methods based on *Hoeffding Trees* may prove more profitable than online methods.

Figures 4.21 and 4.22 show the results of the comparison of the proposed methods with online *state-of-the-art* approaches for two selected real data streams, on which the relationships similar to those occurring in the case of synthetic data can be observed. When the base classifier is GNB, online bagging-based methods note a significant decrease when the *concept drift* occurs, which is not noticeable when using the HT classifier. We also see that in the *poker-lsn-1-2vsAll* stream, which is much more difficult than the *covtypeNorm-1-2vsAll* stream due to a large number of *concept drifts*, online methods employing decision trees perform better than batch methods. Similar dependencies can be observed for the three difficult streams from the INSECTS set presented in Figure 4.23. In the case of GNB, the proposed DSEISC-SB framework performs better than online reference methods. When the base classifier is *Hoeffding Tree*, the reference methods turn out to be better than DSEISC-SB in the case of *sudden* and *gradual concept drift*. For the *incremental concept drift*, the results of the proposed method are comparable with those of the reference methods.

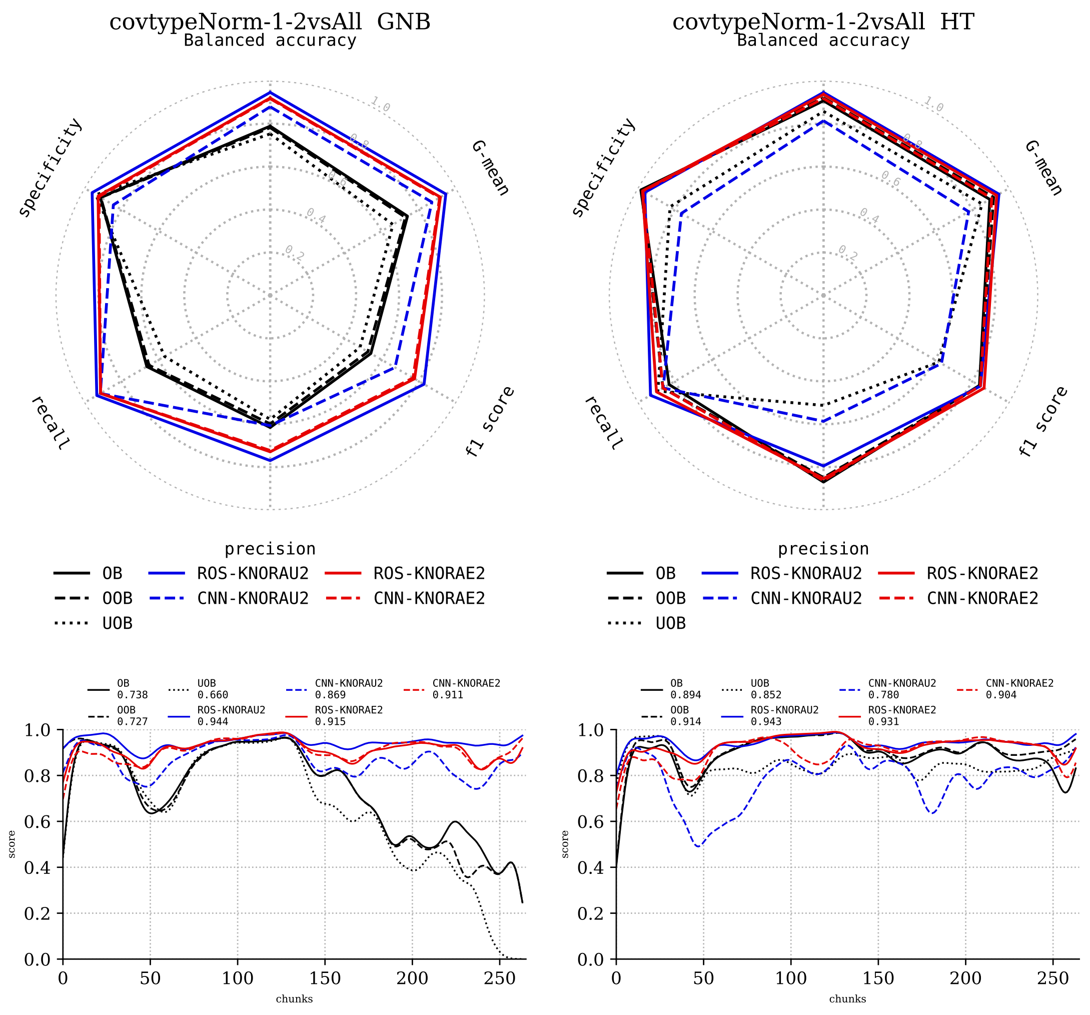


Figure 4.21: *Results of the experiment regarding online reference methods for covtypeNorm-1-2vsAll.*

It is worth noting that, while batch methods using GNB and HT achieved very similar results in the case of synthetic streams, this is no longer the case with real data. It can be seen that methods employing HT as the base classifier note a larger decrease in predictive ability as the concept drift occurs. This may be due to trees being overfitted because of the greater number of instances in each data chunk (1000 instances for real streams and 250 for synthetic streams).

Chunk-based reference methods

Figure 4.24 shows the results of the comparison of the proposed methods with reference *state-of-the-art* chunk-based approaches. As the base classifier, *Gaussian Naïve Bayes* was employed, as in its case, the use of a batch framework based on preprocessing and dynamic classifier selection is more justified than in the case of *Hoeffding Trees* (as shown in Figures 4.19 and 4.20).

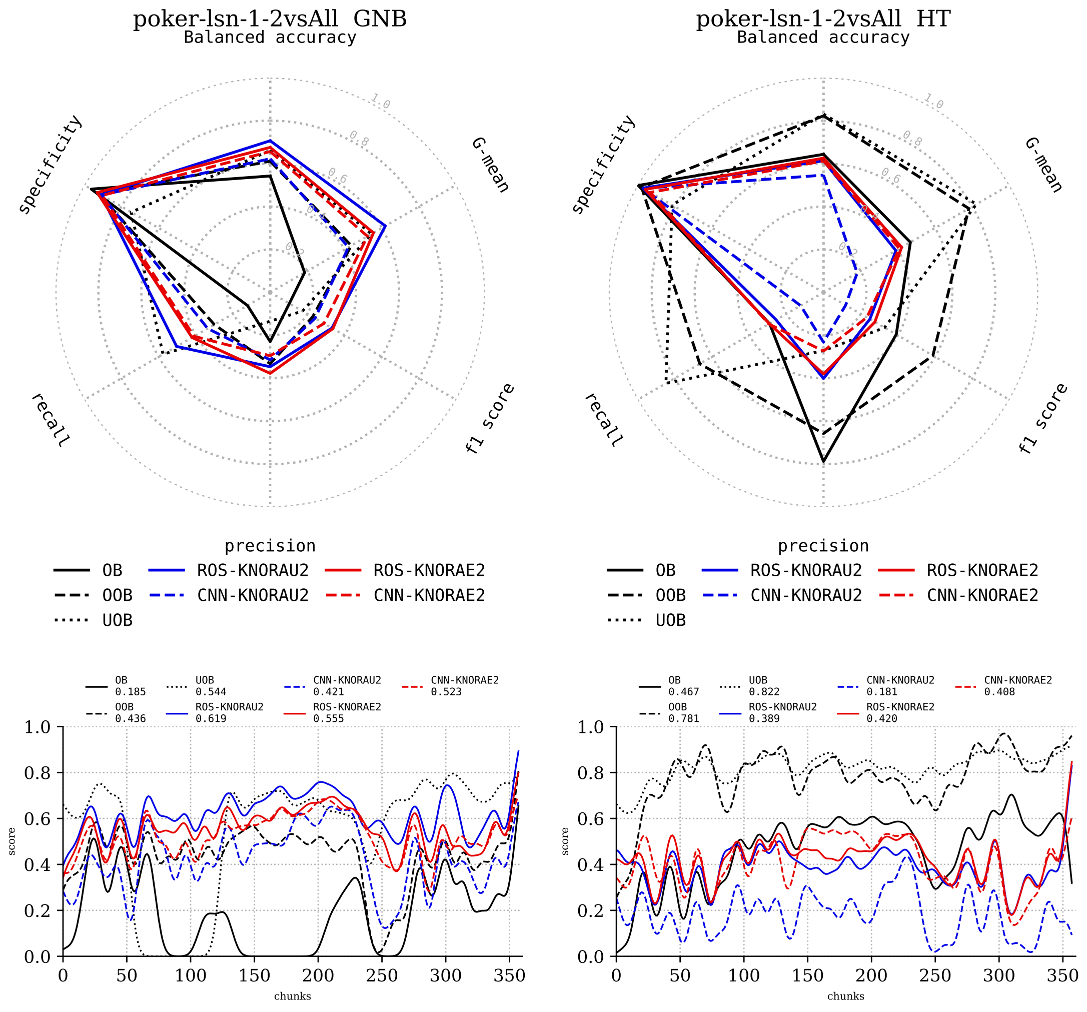
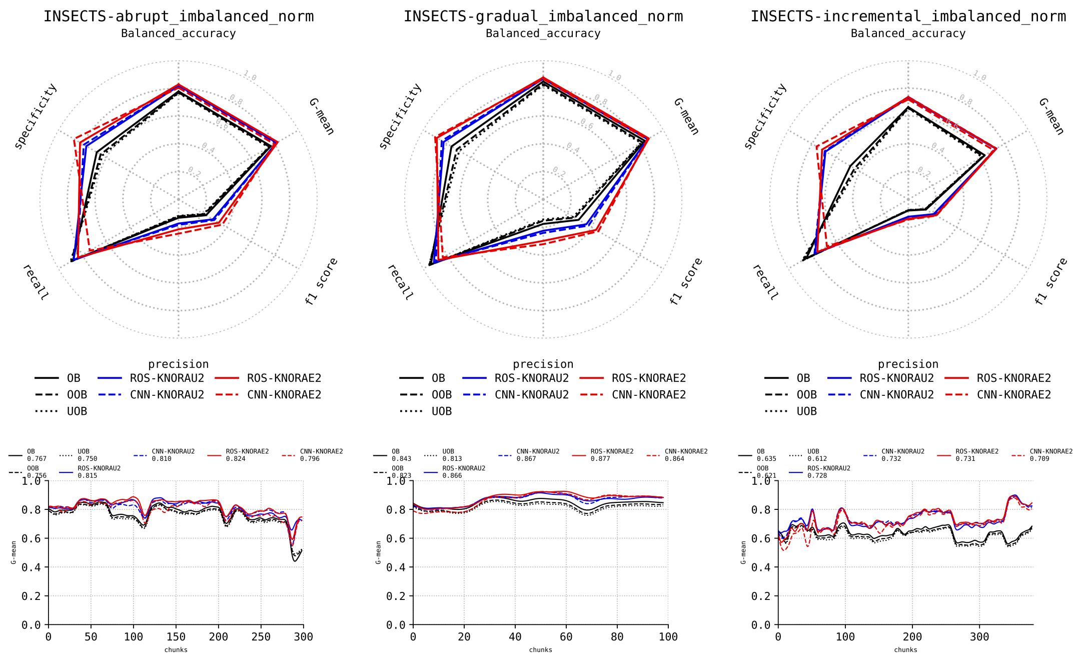


Figure 4.22: *Results of the experiment regarding online reference methods for poker-lsn-1-2vsAll.*

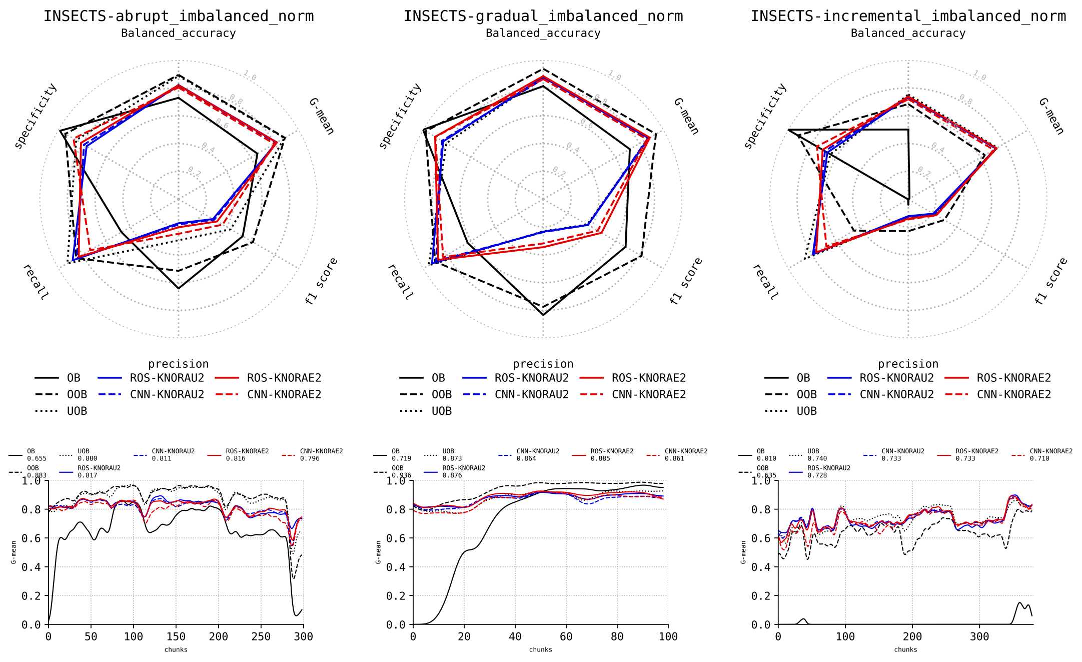
In each case, both for different *Imbalance Ratio* values and for different types of concept drift, the REA method performs by far the worst, obtaining the lowest values of reported metrics (except *specificity*), and also has the most significant decrease at the time of *concept drift* occurence, from which it rises very slowly.

The OUSE approach is the best in the case of the highest tested *Imbalance Ratio* (3% of minority class) and is distinguished by a high *recall* that is achieved at the cost of low *specificity* and *precision*. Although it displays the capacity to cope with the concept drift occurrence, OUSE performs worse as the *Imbalance Ratio* decreases.

Among the reference methods that can be compared with the approaches proposed in this work are *Learn++*.NIE, *Learn++*.CDS and KMC. Regardless of the *Imbalance Ratio* and type of drift, they exhibit behavior comparable to the proposed framework. This is especially true for *Learn++*.CDS, which performs particularly well for the highest *Imbalance Ratio* studied in this experiment, in which in terms of $Gmean_s$ it beats all proposed



(a) GNB

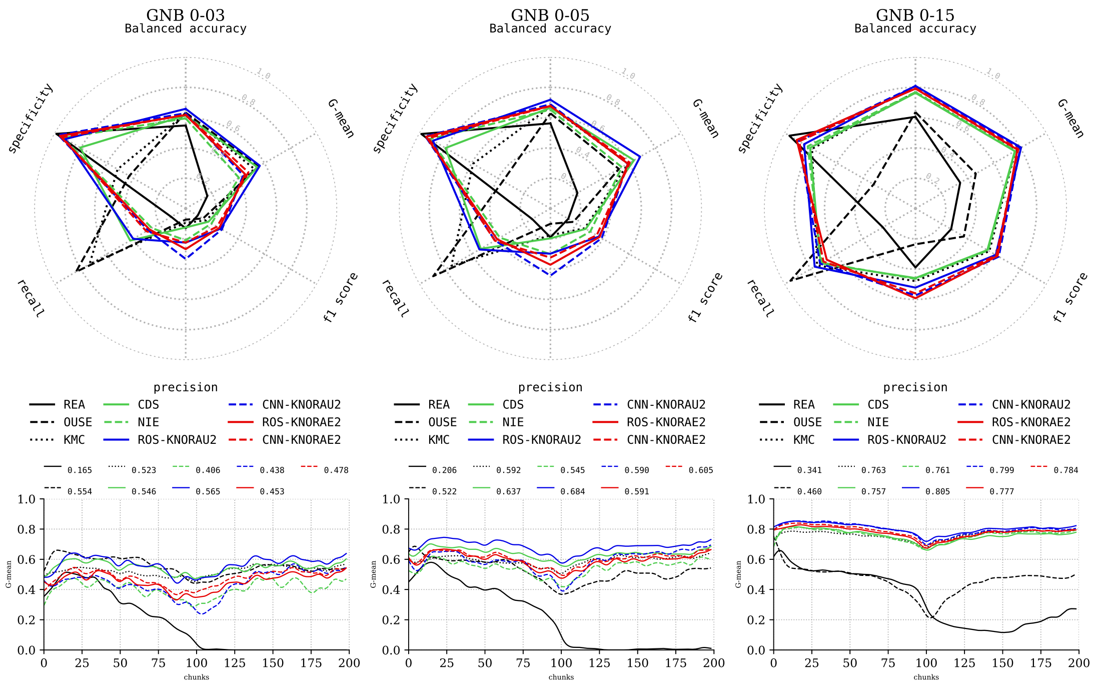


(a) HT

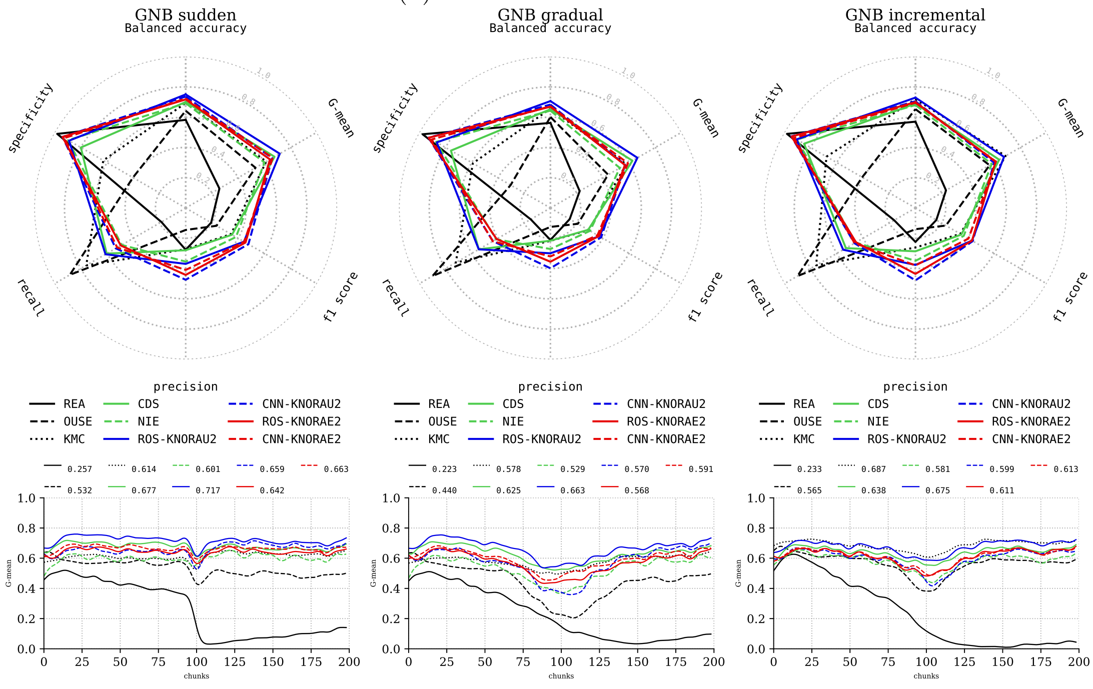
Figure 4.23: *Results of the experiment regarding chunk-based reference methods for the* Gaussian Naïve Bayes *classifier.*

methods except ROS-KNORAU2, while noting a low *precision* value and thus F_1 *score*. It is worth noting that with the decreasing *Imbalance Ratio*, *Learn++*.NIE and *Learn++*.CDS appear to deteriorate compared to methods combining DES and preprocessing techniques.

The KMC method behaves similarly to *Learn++*.CDS, but achieves lower *specificity* and higher *recall*. It performs particularly well in terms of $Gmean_s$ and BAC in the case



(a) Imbalance ratios



(b) Concept drift types

Figure 4.24: *Results of the experiment regarding chunk-based reference methods for the* Gaussian Naïve Bayes *classifier.*

of the incremental drift occurrence in the data stream, where it achieves metric values comparable with the best of the proposed methods (i.e. ROS-KNORAU2). At the same time, however, it displays lower F_1 *scores* than approaches employing *Dynamic Classifier Selection*.

Figures 4.25, 4.26 and 4.27 show results comparing the performance of the proposed methods with *state-of-the-art* batch-based approaches for real data streams. The results

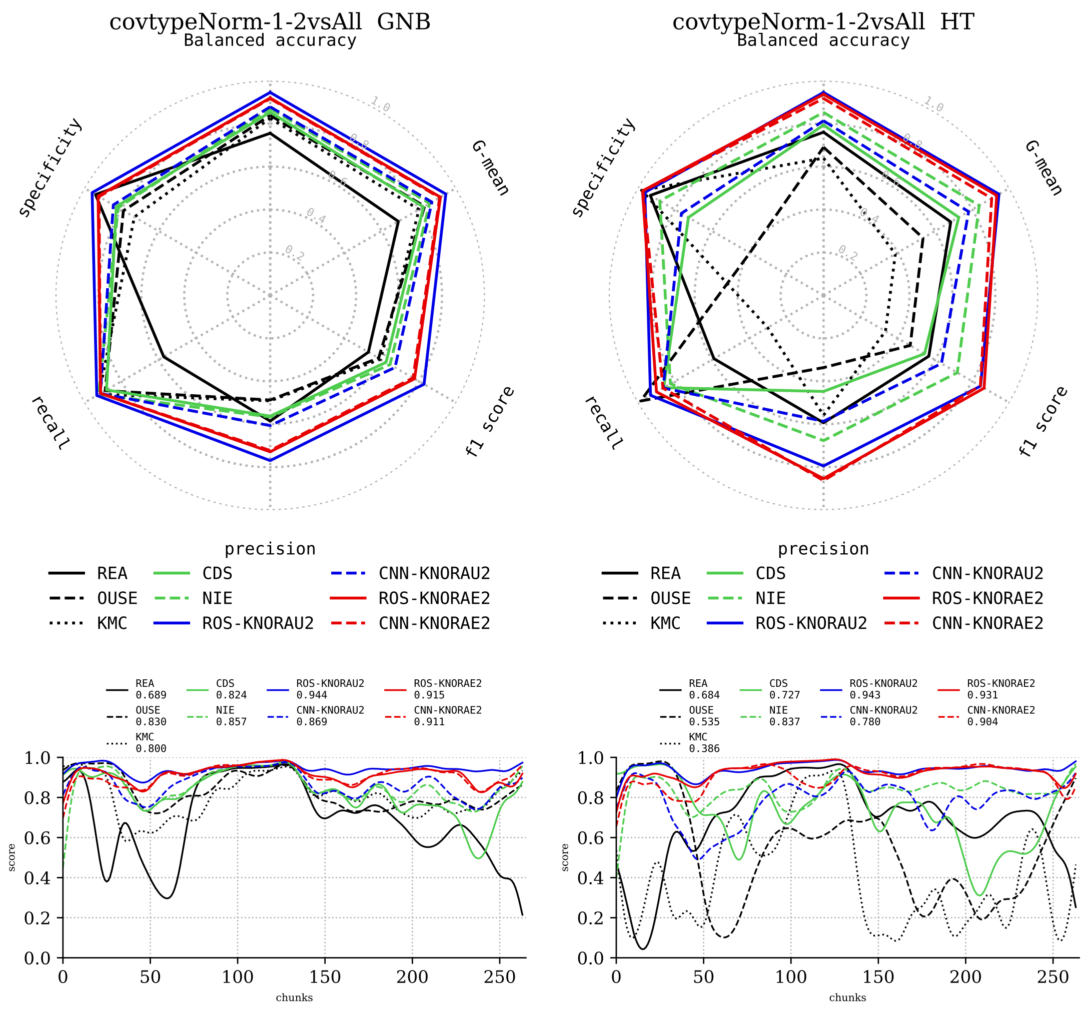


Figure 4.25: *Results of the experiment regarding chunk-based reference methods for covtypeNorm-1-2vsAll.*

obtained coincide with the observations drawn on the basis of experiments carried out on synthetic data streams. The proposed approaches combining preprocessing and DES achieve better results than comparative methods and are more stable. Again, the use of HT classifier for batch methods at chunk size 1000 size leads, especially in the case of more difficult data sets, to deterioration of classification quality and stronger reactions to the occurrence of concept drift.

Observations

Based on the conducted experiments, it can be seen that the results for the methods of batch data stream processing were almost identical for artificially generated streams when the base classifiers were *Gaussian Naïve Bayes* and *Hoeffding Tree*, and each chunk contained 250 samples. The difference between these two base classifiers can bee observed

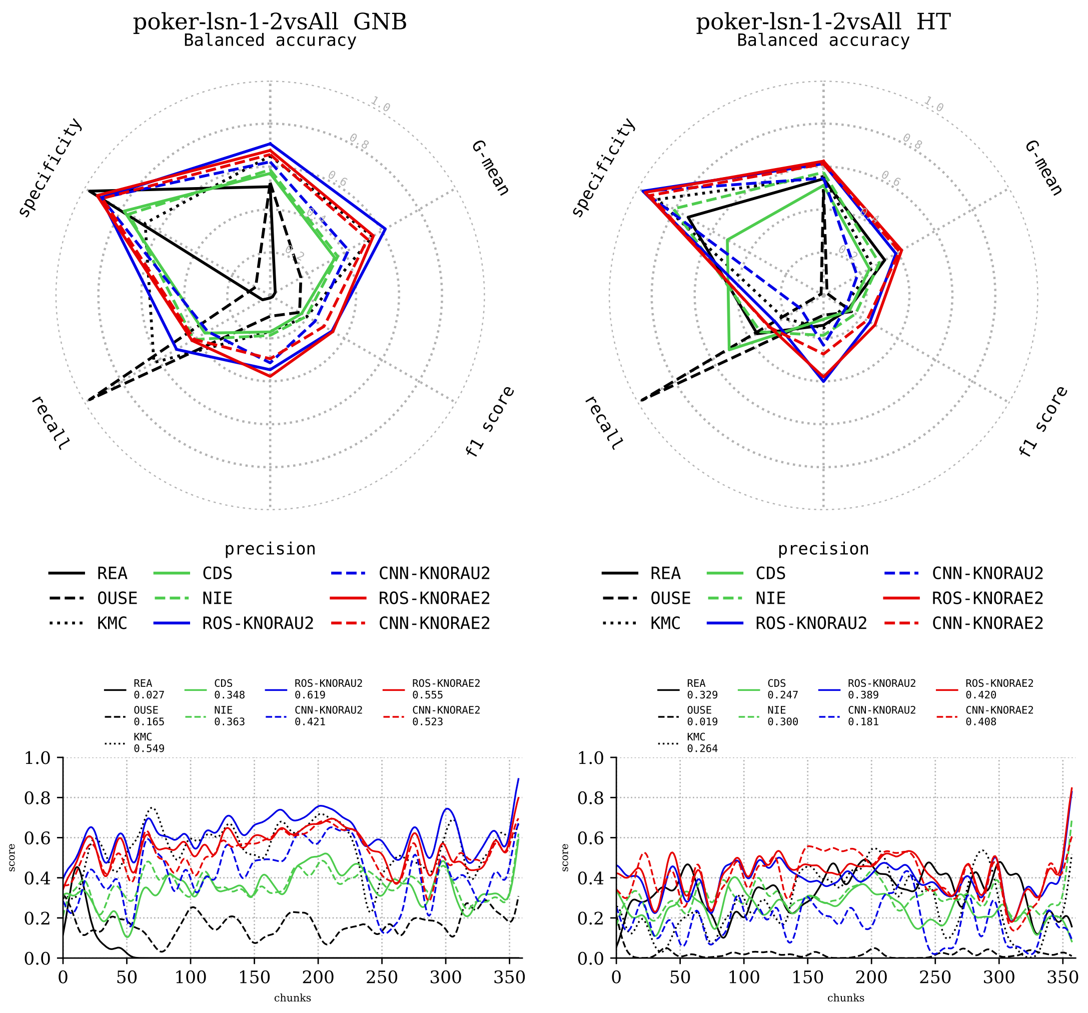
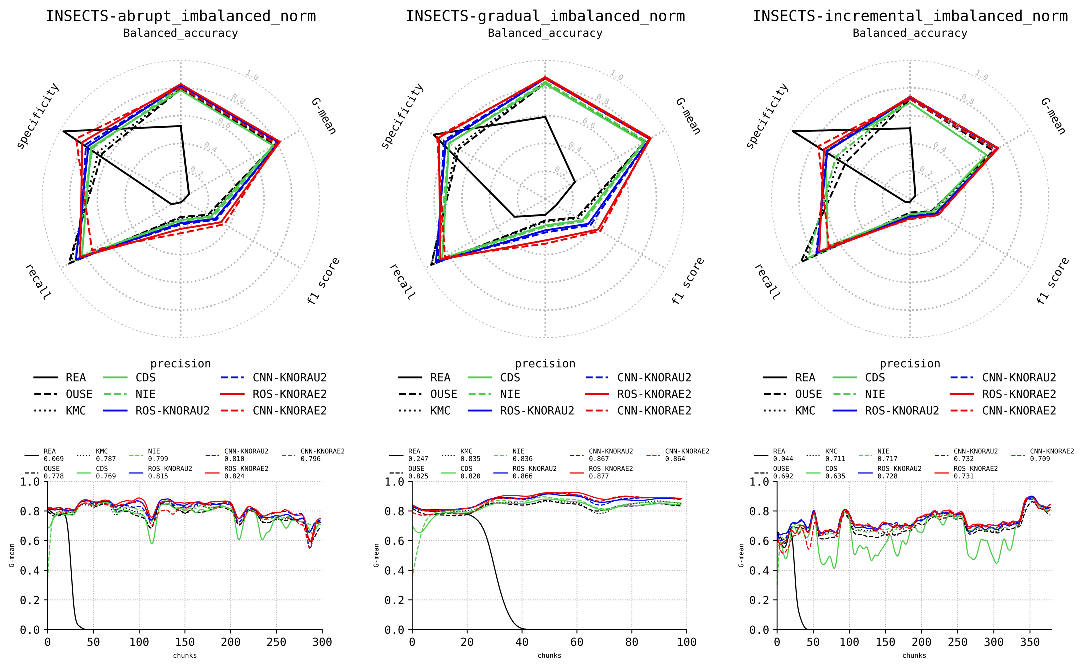


Figure 4.26: *Results of the experiment regarding chunk-based reference methods for poker-lsn-1-2vsAll.*

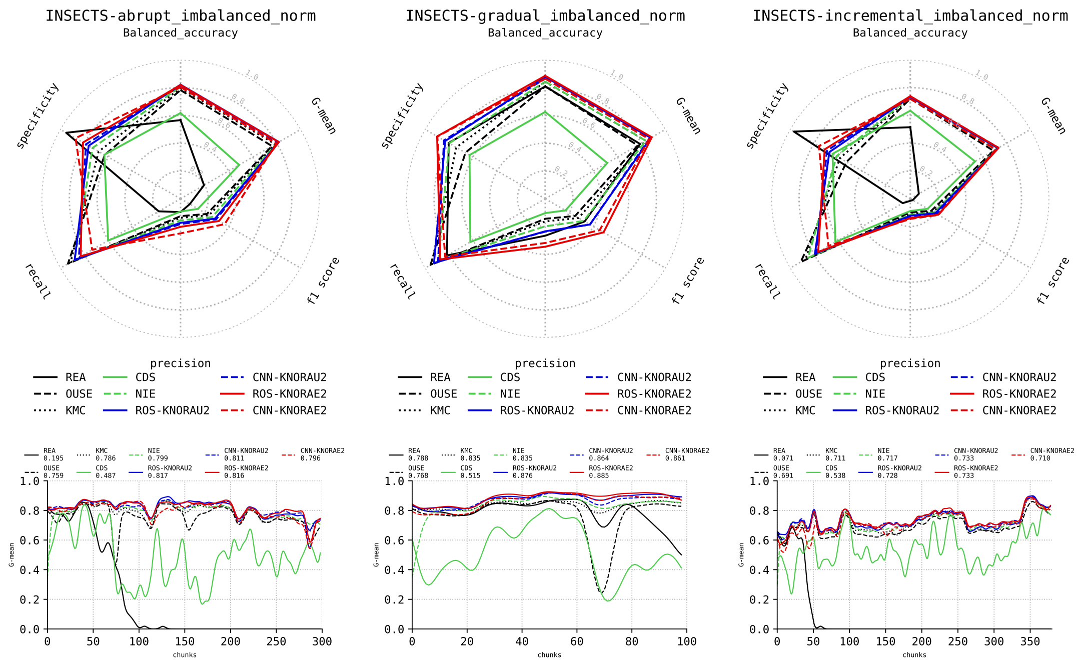
in the case of real data streams when the fixed chunk size was 1000. This may be due to the overfitting of the decision trees.

It can be observed that the use of the dynamic selection method KNORA-E allows the proposed framework for faster restoration in the event of concept drift (especially sudden). This is particularly evident in Experiment 1, in which any preprocessing technique has not been used. These observations have been confirmed by *performance loss* and *restoration time* measures and that was most likely due to the fact that this approach to DES allows for selecting only the classifiers learned on the new concept as soon as in the second data chunk of its presence.

When the SVM was used as the base classifier for the proposed framework, the selection at the level of base models of all bagging sub-ensembles led to a significant deterioration of the achieved results. This may be due to the large pool of not diverse classifiers



(a) GNB



(a) HT

Figure 4.27: *Results of the experiment regarding chunk-based reference methods for the* Gaussian Naïve Bayes *classifier.*

and suggests that in the case of SVM, stratified bagging may not be a good method to diversify individual base classifiers.

The combination of SVM with oversampling in both cases led to a deterioration in its performance compared to the version without preprocessing. *Borderline2*-SMOTE, due to its characteristics, shifted the decision boundary in favor of the minority class, leading to a decrease in *precision*. *Random oversampling*, on the other hand, significantly worsened the results achieved in terms of all measured metrics, except for *specificity*, because duplicate minority class instances resulted in a strong shift of the decision boundary.

When undersampling methods were employed, the use of *Random undersampling* allowed for a more accurate adjustment of the decision boundary and thus a significant improvement in *recall*, F_1 *score*, $Gmean_s$ and BAC at the expense of hindering *precision*. The use of CNN resulted in a similar behavior as in the absence of preprocessing. This is due to the internal structure of this undersampling method, which leaves instances close to the decision boundary, and thus leads to only minor changes.

When it comes to online methods (i.e., OB, OOB and UOB), the use of the *Gaussian Naïve Bayes* classifier leads to a significant deterioration of methods at the moment of *concept drift* occurrence and difficulties with recovering after the drift. OOB and UOB mitigate these effects due to built-in resampling mechanisms, but they still struggle due to the fact that GNB remembers the previous concept.

Decision trees do much better in online methods because they have the opportunity to achieve optimal predictive ability (as seen before *concept drift* occurs) and they also cope better with recovery after drift. Generally, when *Hoeffding Trees* are used, online methods work better than the proposed batch methods, except for the moment when *sudden drift* occurs, in which case online methods rebuild more slowly than chunk-based ones because the classifiers trained on the old concept are not removed. Theoretically, with many *sudden drifts* in a single data stream, chunk-based methods can have an advantage over online ones, even when using decision trees as base estimators.

REA is by far the worst-performing one of the chunk-based reference methods, especially in the case of *concept drift* occurrence. In this approach, added to the training sets are minority class samples from the old concept, which makes it difficult for the method to recover after drift. Besides, all models are subject to weighted combination, as there is no forgetting mechanism.

OUSE builds a new ensemble on each of the data chunks so that the classifiers relate to the current concept. Despite using all minority class instances that have ever appeared in the stream, the profit from balancing the problem using real samples, in the case of a high *Imbalance Ratio*, outweighs the loss resulting from using some of the instances from the old concept. However, as the *Imbalance Ratio* decreases, and thus the number of instances from the old concept in training set increases, the algorithm begins to deteriorate.

The KMC method presents an approach similar to the ones proposed in DESISC and DESISC-SB. As a base, it uses the SEA algorithm in which the ensemble is pruned using the AUC metric. Additionally, it uses the undersampling method based on the *k-Means* algorithm. It achieves particularly promising results in case of the *incremental concept drift* occurrence. This may be due to the fact that it does not use real majority class instances but its clusters centroids that might better reflect the slowly occurring minor concept changes.

Learn++.DCS performs comparatively with the proposed methods in terms of $Gmean_s$ and BAC, but at the expense of *precision* and F_1 *score*. It is somewhat comparable to the proposed framework, as it also uses preprocessing, but the *Dynamic Classifier Selection* is replaced by a weighted combination. However, the method deteriorates compared to those proposed as the *Imbalance Ratio* decreases. *Learn++*.NIE is also quite similar to the proposed framework in that it uses the bagging sub-ensembles that train each of the base classifiers on the whole minority class from the given data chunk and part of the majority class. It is done in such a way, that no information about the majority class is lost. Sub-ensembles are then integrated utilizing the *recall*-based weighted combination. In the case of both methods, the main difference between them and the proposed framework is the use of dynamic selection, which seems to perform better than the weighted combination.

Answers to research questions

The answers to the previously formulated research questions are as follows:

Q1. Which *Dynamic Ensemble Selection* methods perform best while dealing with the *concept drift* occurrence?

A1. Based on the results obtained in Experiment 1, it can be concluded that KNORA-E is the *Dynamic Selection* method that best copes with the *concept drift* phenomenon. This is due to the approach to classifier selection that prefers only local oracles, which allows for quick recovery of the generalization ability after the *concept drift* occurrence.

Q2. Does performing *Dynamic Ensemble Selection* at the level of all generated base models (including those forming individual bagging classifiers) allow *dseisc-sb* to achieve better performance when compared to *Dynamic Selection* performed only at the level of bagging classifiers?

A2. Conducted experiments confirmed, that performing *Dynamic Ensemble Selection* at the level of all generated base models leads to better performance when compared to *Dynamic Selection* performed only at the level of bagging classifiers. This is due to the larger and more diverse pool of available models.

Q3. Can methods combining data preprocessing and *Dynamic Ensemble Selection* outperform *state-of-the-art* batch-based and online classifiers for difficult data stream classification task?

A3. The results obtained in Experiment 3 confirmed, that the DSEISC-SB framework may outperform both bath and online-based *state-of-the-art* imbalanced data stream classification algorithms.

Chapter 5

Limited access to labels

A significant problem when building classifiers based on data stream is information about the correct label. Most algorithms assume access to this information without any restrictions. Unfortunately, this is not possible in practice because the objects can come very quickly and labeling all of them is impossible, or we have to pay for providing the correct label (e.g., to human expert). Hence, methods based on partially labeled data, including methods based on an active learning approach, are becoming increasingly popular, i.e., when the learning algorithm itself decides which of the objects are interesting to improve the quality of the predictive model effectively.

This chapter introduces the new method for active learning of data stream classifier. The BALS algorithm in based on the notion, that the classifier should receive - in addition to selected labeled objects by the active learning strategy - a pool of randomly selected objects from each data chunk.

Then, the behavior the DESISC-SB framework combining DES and preprocessing for imbalanced data stream classification under the limited access to labels scenario is evaluated. Best performing variant of DESISC-SB is coupled with random labeling and active learning strategy in order to see what is the effect of limited labeling on ensemble methods for imbalanced data stream classification.

5.1 *Budget Active Labeling Strategy*

The problem of limited label access is important due to its prevalence in real data. When dealing with data streams the problem is not only the cost of obtaining labels but also the speed at which the data arrives, which may prevent labeling the number of samples that would allow the model to achieve the expected classification performance. Recent works in this field noticed that in the event of rapid changes, using labeling strategies only for data close to decision boundaries may not be enough to adapt the classifier to the new distribution sufficiently (especially in the case where the changes in the distributions are very significant) [134]. Therefore, this section proposes that the classifier should receive, in addition to selected objects labeled by the active learning strategy, a pool of randomly selected instances from each data chunk. This proposition is called *Budget Active Labeling Strategy* (BALS).

BLS

The research presented in this work is based on three approaches to classifiers' building on streaming data with limited labeling. The first of them is, hereinafter referred to as the BLS, *Budget Labeling.* In BLS, for each data chunk, the actual labels are obtained for a fixed percentage of randomly selected samples, denoted by the *budget* parameter b. This approach is presented in Algorithm 12. The description of the functions used in the pseudocode is as follows:

- RANDOMBUDGET() – selects, according to the set budget b, a fixed number of problem instances randomly chosen from current data chunk $\mathcal{DS}_k$.
- GETLABELS() – obtains the real labels for previously selected samples and constructs a learning set $\mathcal{LS}_k$.
- UPDATECLASSIFIER() – updates the classifier Ψ with learning set $\mathcal{LS}_k$.

Algorithm 12 Pseudocode for BLS

```
Input:
    $Stream = \{\mathcal{DS}_1, \mathcal{DS}_2, \dots, \mathcal{DS}_k, \mathcal{DS}_{k+1}, \dots\}$ – data stream,
    $\Psi$ – classification algorithm,
    $b$ – budget value.

1: for each $k, \mathcal{DS}_k = \{x_k^1, x_k^2, \dots, x_k^N\}$ in $Stream$ do
2:     $\mathcal{X}_k$ = RANDOMBUDGET($b, \mathcal{DS}_k$)        ▷ Randomly select percentage of instances
3:     $\mathcal{LS}_k$ = GETLABELS($\mathcal{X}_k$)               ▷ Get labels for the chosen instances
4:     $\Psi \leftarrow$ UPDATECLASSIFIER($\Psi, \mathcal{LS}_k$)          ▷ Update the classifier
5: end for
```

ALS

The second approach is a simple *active learning* solution, further described by the ALS acronym and presented in Algorithm 13. In the case of this method, after incrementally training the model on the fully labeled first data chunk (steps 2 and 3), the processing of each subsequent one begins with collecting the support of the existing model Ψ (which forces the application of probabilistic classifier) obtained for the current data chunk $\mathcal{DS}_k$. The objects are later sorted according to the distance from the decision boundary, which for a binary problem means an absolute difference from the value of .5. Real labels are obtained for objects for which the calculated absolute difference does not exceed the set threshold t (steps 5 and 6). In the pseudocode, the one new function was used:

- ACTIVELEARNING() – selects, according to the set threshold t, all problem instances for which the distance from the decision boundary doe not exceed the set value.

Algorithm 13 Pseudocode for ALS

```
Input:
    $Stream = \{\mathcal{DS}_1, \mathcal{DS}_2, \ldots, \mathcal{DS}_k, \mathcal{DS}_{k+1}, \ldots\}$ – data stream,
    $\Psi$ – classification algorithm,
    $t$ – threshold value.

1: for each $k, \mathcal{DS}_k = \{x_k^1, x_k^2, \ldots, x_k^N\}$ in $Stream$ do
2:     if $k == 0$ then
3:         $\Psi \leftarrow$ UPDATECLASSIFIER($\Psi, \mathcal{DS}_k$)   ▷ Update the classifier using whole chunk
4:     else
5:         $\mathcal{X}_k =$ ACTIVELEARNING($t, \mathcal{DS}_k$)   ▷ Select instances using active learning
6:         $\mathcal{LS}_k =$ GETLABELS($\mathcal{X}_k$)   ▷ Get labels for the chosen instances
7:         $\Psi \leftarrow$ UPDATECLASSIFIER($\Psi, \mathcal{LS}_k$)   ▷ Update the classifier
8: end for
```

BALS

The *Budget Active Labeling Strategy* algorithms, which is the main contribution of this section, combines both the *Budget Labeling* and *active learning* approaches described in Algorithms 12 and 13. It uses an active strategy, typical for ALS (step 5), but each performed active selection of objects is supplemented by a certain, predetermined random samples pool, like in BLS strategy (step 6). The proposed approach thus tries to increase the generalization ability of the used classification algorithm, by additional diversification of samples subjected to labeling by an expert.

Computational and memory complexity analysis

The BLS algorithm uses a simple sampling without replacement in order to choose the

Algorithm 14 Pseudocode for BALS

```
Input:
    $Stream = \{\mathcal{DS}_1, \mathcal{DS}_2, \ldots, \mathcal{DS}_k, \mathcal{DS}_{k+1}, \ldots\}$ – data stream,
    $\Psi$ – classification algorithm,
    $t$ – threshold value,
    $b$ – budget value.

1: for each $k, \mathcal{DS}_k = \{x_k^1, x_k^2, \ldots, x_k^N\}$ in $Stream$ do
2:     if $k == 0$ then
3:         $\Psi \leftarrow$ UPDATECLASSIFIER($\Psi, \mathcal{DS}_k$)  ▷ Update the classifier using whole chunk
4:     else
5:         $\mathcal{X}_k =$ ACTIVELEARNING($t, \mathcal{DS}_k$)  ▷ Select instances using active learning
6:         $\mathcal{X}_k \leftarrow$ RANDOMBUDGET($b, \mathcal{DS}_k$)  ▷ Randomly select percentage of instances
7:         $\mathcal{LS}_k =$ GETLABELS($\mathcal{X}_k$)  ▷ Get labels for the chosen instances
8:         $\Psi \leftarrow$ UPDATECLASSIFIER($\Psi, \mathcal{LS}_k$)  ▷ Update the classifier
9: end for
```

random budget b of instances from each data chunk $\mathcal{DK}_k$. This operation has the computational complexity of $O(b\ log\ b)$. The used *active learning* approach calculates each sample's distance from the decision boundary (which an absolute difference of obtained support and .5), which has the complexity of $O(\mid \mathcal{DS}_k \mid)$. Then, ALS sorts the objects according to the acquired distance and uses only those, for which the distance values does not exceed the set threshold t. This operation has the computational complexity of $O(\mid \mathcal{DS}_k \mid log \mid \mathcal{DS}_k \mid)$. The proposed *Budget Active Labeling Strategy* combines both approaches.

5.1.1 Experimental evaluation

This subsection presents the motivation, goals and set-up of the performed experiments, as well as their results.

Research questions

The experiments were designed to answer the following questions:

Q1. Can a classifier that will have a quality comparable to the model learned on all available objects be obtained by using a small budget combined with active learning for data labeling?

Q2. Will such a method be better in terms of the drift response time (restoration time) and performance deterioration, when compared to the reference methods for dealing with limited labeling?

Q3. Will the observed behavior also occur when dealing with imbalanced real data streams?

Goals of the experiments

Experiment 1 – Balanced synthetic data streams

The main purpose of the first experiment is to evaluate the quality of the BALS method, when compared to the MLP trained on all available data, BLS, and ALS for the task of balanced drifting data stream classification.

Experiment 2 – Imbalanced real data streams

The main goal of the second experiment is to observe the behavior of the tested methods, when dealing with the imbalanced real data streams.

Experimental set-up

The analysis was based on six types of synthetic streams, replicated 10 times for stability of the achieved results. The detailed characteristics of the generated streams are described below:

- *Concept drift* types – *sudden*, *gradual* and *incremental*,
- Approaches to repetitive concepts – *recurrent* and *non-recurrent concept drift*,
- Data stream size – 500 000 instances (1000 data chunks, 500 instances each)
- Number of concept drift per stream – 9.

Additionally, during Experiment 2, the proposed method was evaluated on the 5 real data streams described in Table 5.1.

Table 5.1: *Real data streams characteristics.*

Data stream	#Samples	#Features	IR
covtypeNorm-1-2vsAll	266 000	54	4
poker-lsn-1-2vsAll	360 000	10	10
INSECTS-abrupt_imbalanced_norm	300 000	33	19
INSECTS-gradual_imbalanced_norm	100 000	33	19
INSECTS-incremental_imbalanced_norm	380 000	33	19

The three considered methods were implemented in consistency with the *scikit-learn* [187] API. Evaluation was based on 7 different metrics and performed according to the *Test-Then-Train* methodology. The details on experimental set-up are listed below:

- Classification algorithm – incremental MLP probabilistic classifier with *ReLu* activation function, *Adam* solver and one hidden layer consisting of 100 artificial neurons,
- Methods' parameters:
  - ALS – the budget of 5, 10 and 20%,
  - BLS – a single threshold of absolute distance $t = .2$ from the decision boundary,
  - BALS – threshold of absolute distance $t = .2$ as well as the random budget of 5, 10 and 20%,
- Evaluation metrics:
  - Experiment 1 – *accuracy score*,
  - Experiment 2 – BAC, F_1 *score*, $Gmean_s$, *recall*, *precision*, and *specificity*,

Experiments can be replicated according to the code available on the *GitHub* repository[1].

Experiment 1 – Balanced synthetic data streams

The experimental studies were carried out for three different *concept drift* types. Figure 5.1 presents the runs for individual approaches to the construction of the MLP-based model for data containing *sudden* drifts.

As can be seen, the BLS approach to *non-recurring sudden* drifts is characterized by a constant learning curve that builds the model in a similar way for each of the following concepts. The learning curve achieved has lower dynamics than the full model (marked with dotted lines) and in no case reaches the maximum generalization capability. The lower learning dynamic is directly caused by the reduction of the number of learning objects. Interestingly, there are no significant differences in quality between using 5, 10 or 20% of objects.

For recurrent drifts, there is a slight change in the behavior of the BLS approach. Achieving full discriminative ability causes the model to retain information from previous concepts even after they have been changed. While in the case of the first *concept drift*, which introduces a new distribution of problem classes for the first and only time, quality degradation occurs in the same way as in the full model when a sudden change occurs. In other situations, the quality reduction is less noticeable. However, it does not decrease

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

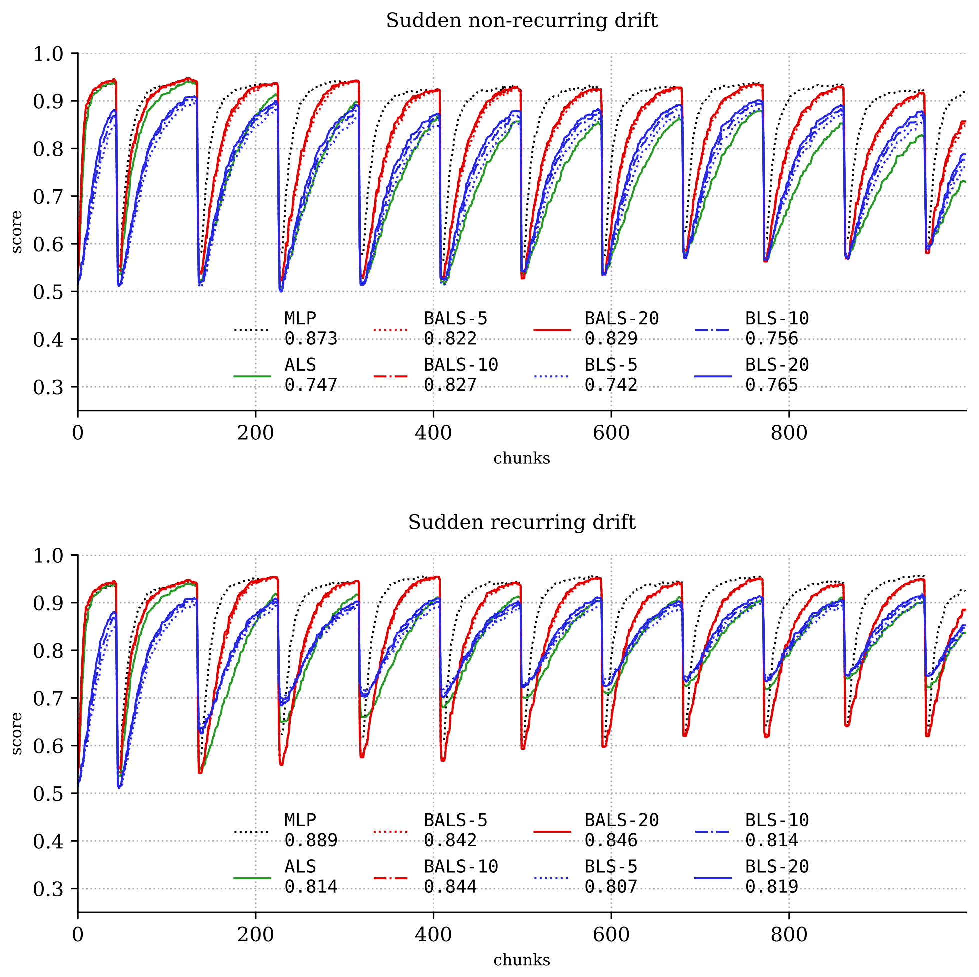
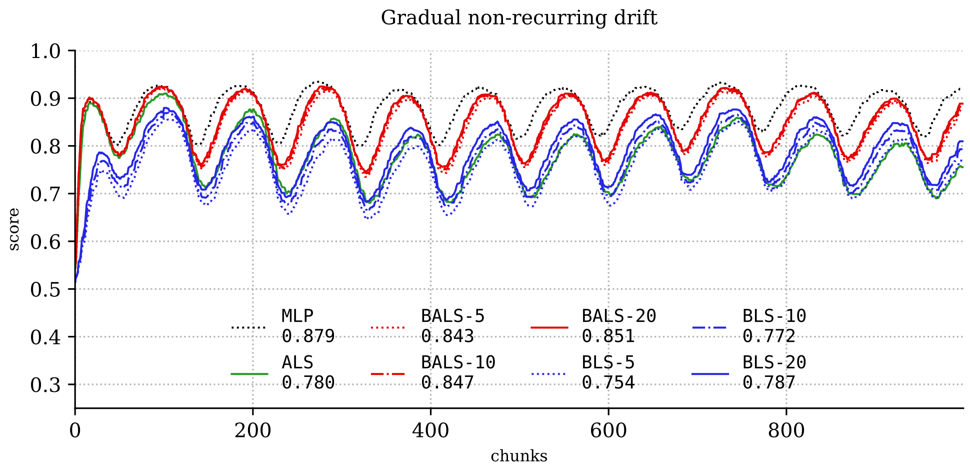


Figure 5.1: *Exemplary results for the stream affected by a sudden concept drift.*

significantly in subsequent iterations of *recurrent* drifts, and the other models achieve higher classification quality than BLS relatively quickly each time.

The most interesting observation in this case is the behavior of the classification approach ALS. While in the case of the first and second concepts (regardless of concept repetition) its ability to achieve the full possible classification accuracy (relative to MLP trained on a fully labeled data chunk) can be seen, its progressive degeneration with subsequent drifts is equally visible. In the case of *non-recurring* drifts, it corresponds with the occurrence of the third concept BLS and decreases in accuracy over time. In the case of *recurrent* drifts, this degeneration occurs even faster due to the previously described remembering of old concepts by BLS and already with the occurrence of the second *concept drift* it turns out that ALS is outperformed by the competitor based on a random budget.

The observation of the BALS method for the first two concepts is identical to the ALS approach, and in both cases leads to the achievement of the generalization ability of the classifier built on fully labeled data. However, it is pleasantly surprising that the



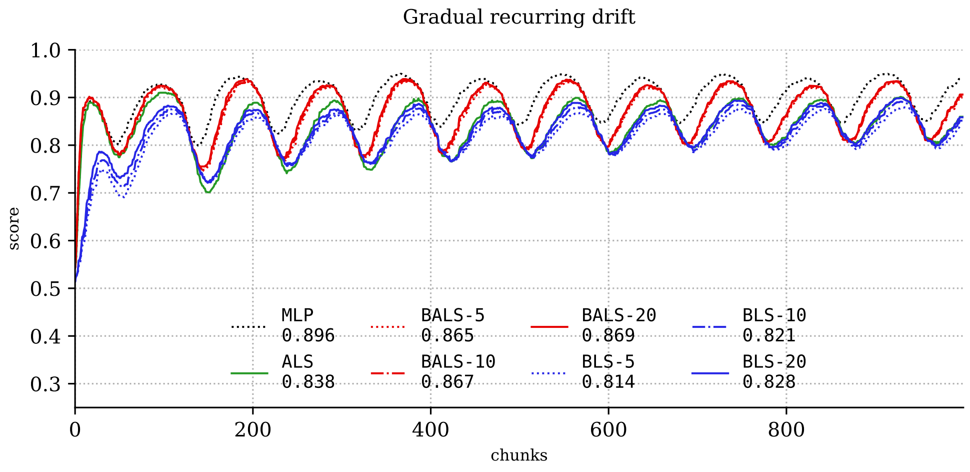
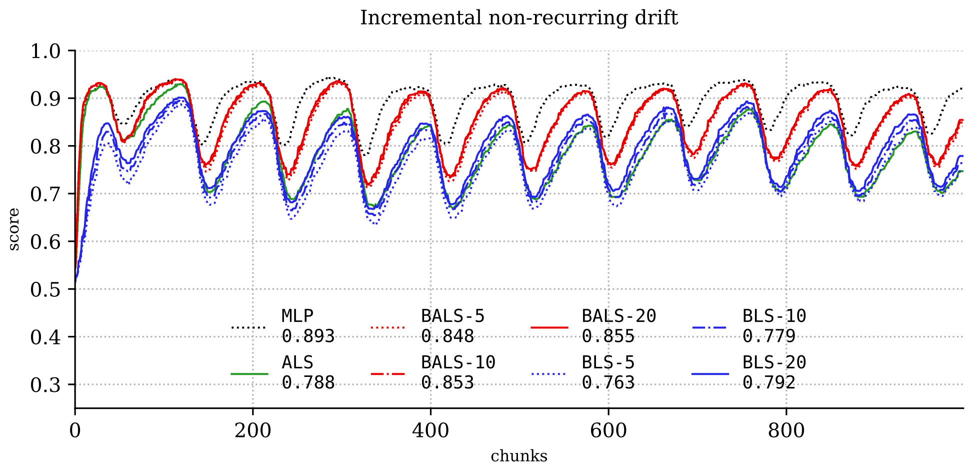


Figure 5.2: *Exemplary results for the stream affected by a gradual concept drift.*

introduction of even a small percentage of random patterns sensitizes such a method to the degeneracy of subsequent drifts typical of BLS. The difference between the two standard approaches (BLS and ALS) and the combined approach is not just a simple improvement in classification accuracy. It can be seen that by introducing randomly selected patterns, the BALS method achieves the full possible classification accuracy every time (albeit sometimes with decreasing dynamics).

The proportional to learning time degeneration of the ALS approach is probably due to the increasing certainty of the predictions made, in the case of analysis of the supports achieved, which means their strong polarization, and thus a gradual, rapid reduction of the number of objects located near the decision boundary. This means that the solution based on support thresholding – over time – assigns fewer and fewer objects as potentially useful for labeling. The phenomenon of this polarization is reduced by introducing seemingly different patterns for the built recognition model, modifying the



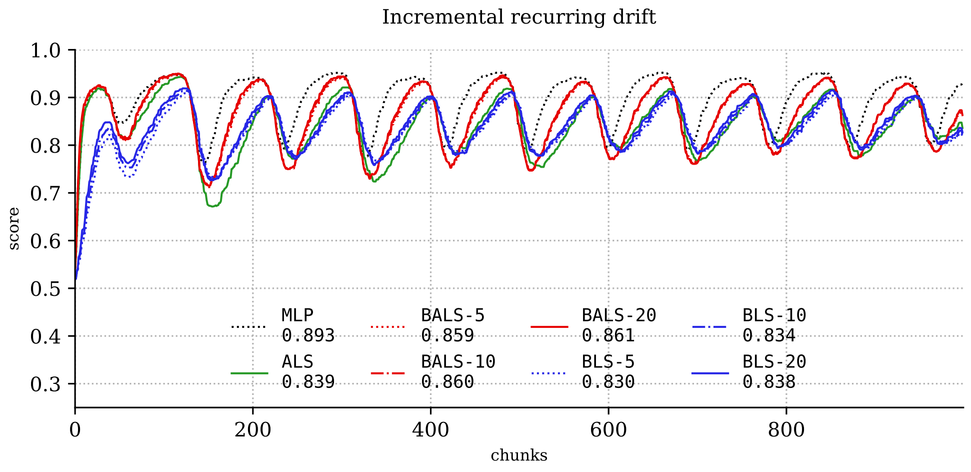


Figure 5.3: *Exemplary results for the stream affected by an incremental concept drift.*

statistical distribution of obtained support, which is a direct result from the new concept *signaling itself* for a need for increased learning rate.

Interestingly, the percentage of random patterns added to the active labeling model does not appear to have a significant impact on classification accuracy or learning curve dynamics. Even a small number of such objects (5%) causes BALS to no longer exhibit the degenerative tendencies of the pure ALS model.

The observations made for *sudden* drifts, including both approaches to drift recurrence, can be directly applied to those made for *gradual* (Figure 5.2) and *incremental* drifts (Figure 5.3). The dynamics of the concept changes themselves do not seem to have much influence on the relationships between the analyzed algorithms, so the conclusions made for *sudden* drift can be easily generalized for all problems considered in the research.

Experiment 2 – Imbalanced real data streams

Figure 5.4 shows the behavior of the proposed MLP model construction approaches when classifying real imbalanced data streams. The radar diagrams show the average values of all six analyzed metrics, while the runs are presented for the $Gmean_s$.

In the case of the CovType stream, which has the lowest imbalance ratio of all the real data streams analyzed, there are clear - although of undefined type - concept drifts. The BLS method achieves the generalization capacity of the full model, but the learning curve has lower dynamics than in the case of access to the full training set, which is again due to the lower number of patterns used in the training process. BLS also shows a greater, but delayed in relation to the full model, decrease in generalization ability when the concept drift occurs (degrading to the level of a random classifier), which may be due to the occurrence of drift in prior class probabilities and a temporary increase in the imbalance ratio. When rebuilding the model, the BLS achieves a generation ability close to that of the full model, which increases with the percentage of budget used. The MLP model trained with the use of ALS performs by far the worst, remaining for most of the data stream at the level of the random classifier. Only in the vicinity of chunk 180 does the learning curve begin to be visible, which leads to the achieved generalization ability being close to the full model. It may be caused by too high certainty of the prediction, which translates into the lack of instances located within a fixed distance from the decision boundary. The proposed BALS approach, combining ALS with a random budget, allows to achieve full generalization capacity faster than in the case of BLS, but in the event of concept drift it shows a faster reduction in performance. Reconstruction after drift occurrence in the case of BALS is slower than in the case of BLS, however, a higher value of the examined metrics is achieved.

In the case of the Poker stream, which exhibits a higher imbalance ratio of 10, the BLS - as expected - has the lowest ability to detect the minority class. This is due to the fact that mainly the majority class instances are drawn to the budget. The model trained with the use of ALS, despite a poor start and remaining at the level of the random classifier during the first 150 data chunks, at a later stage of the stream achieves the generalization ability exceeding that achieved by the full model. It is caused by changes in the support space, which lead to an increase in the number of problem samples occurring at a set distance from the decision boundary. The use of the BALS strategy allows the observation of behavior identical to that displayed when dealing with synthetic balanced data streams. In the first half of the stream, the model trained using the BALS approach, as opposed to BLS and ALS, performs above the random classifier level, and in the second half, it achieves the generalizing ability that exceeds that of the full model.

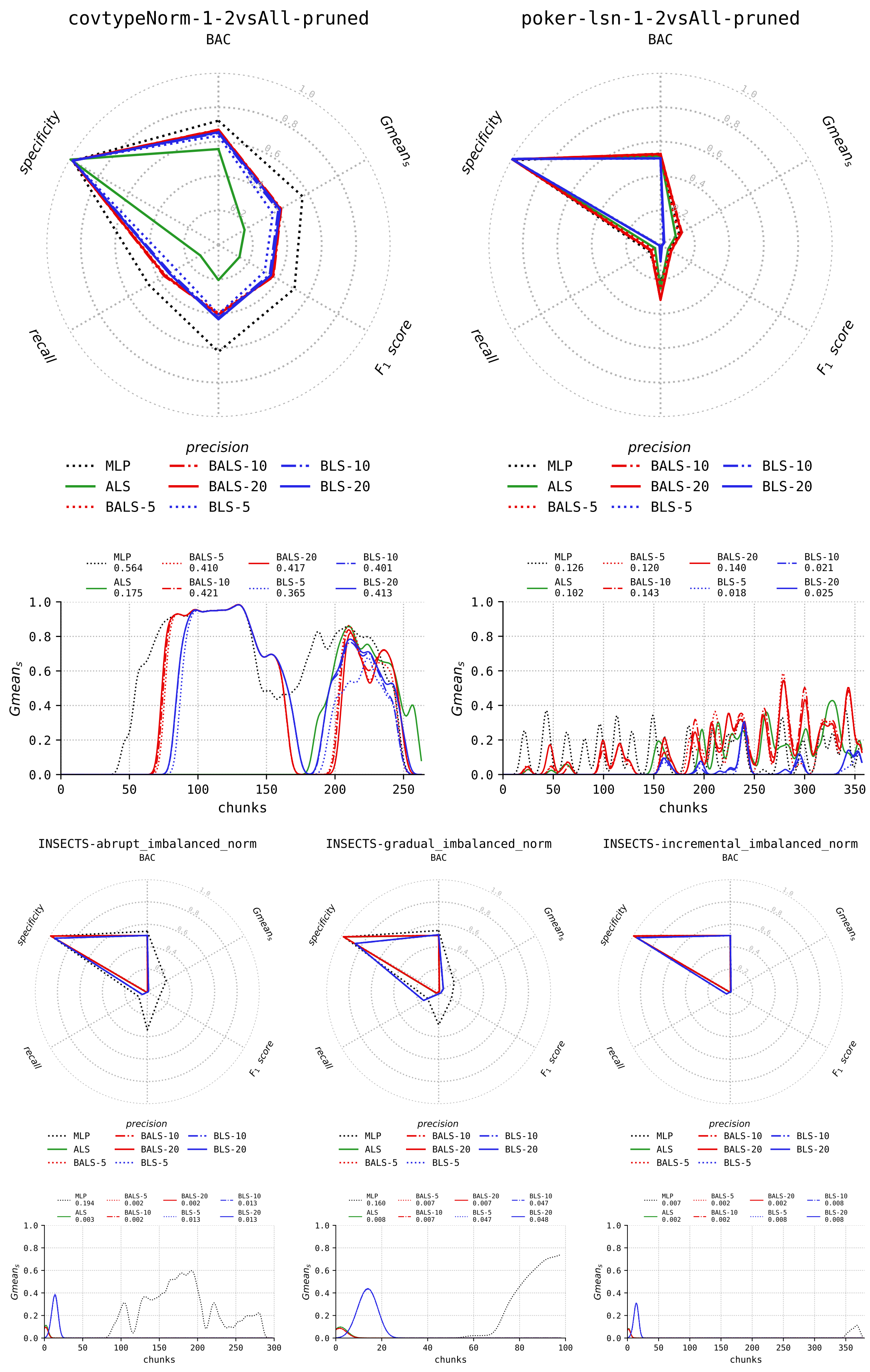


Figure 5.4: *Results for real imbalanced data streams.*

In the case of INSECT data streams, which present by far the most difficult problems and the highest imbalance ratio, the MLP models in combination with the proposed labeling strategies are not able to cope with the classification task. The models' performance only exceeds that of the random classifier at the beginning of all three streams, which may be due to a drift in prior probabilities and a lower imbalance ratio in the first twenty-five data chunks. In the further part of the streams, the limitation of the training set size makes it impossible to achieve the generalization ability above that of the random classifier.

Observations

The BALS outperforms ALS algorithm due to the use of an additional fraction of labeled instances. However, its size was very small compared to the fraction of objects selected according to the active learning rule. Additionally, increasing its number does not significantly improve the quality of the proposed method.

It is obvious that the proposed model obtained slightly worse results compared to the classifier based on a fully labeled learning set, but the time needed to reach the same performance is very short.

Answers to research questions

Q1. Can a classifier that will have a quality comparable to the model learned on all available objects be obtained by using a small budget combined with active learning for data labeling?

A1. Performed experiments confirmed, that the BALS method – combining both random and active labeling – is capable of obtaining a generalization ability at the level of full model.

Q2. Will such a method be better in terms of the drift response time (restoration time) and performance deterioration, when compared to the reference methods for dealing with limited labeling?

A2. The obtained results confirmed, that the model trained using BALS approach may display better restoration time as well as less performance deterioration than BLS or ALS when dealing with *concept drift*.

Q3. Will the observed behavior also occur when dealing with imbalanced real data streams?

A3. The conducted experiments proved, that the BALS method can be successfully used in the case of relatively highly imbalanced data streams, even without the use of additional data preprocessing.

5.2 DESISC-SB framework under limited labels scenario

This section focuses on extending the DESISC-SB imbalanced data stream classification framework with an active learning module. This is to asses the compatibility of the proposed batch approach with active labeling methods and to evaluate its behavior, compared to a single MLP classifier, when dealing with restricted access to labels. The schema of the expanded framework is presented in the Figure 5.5.

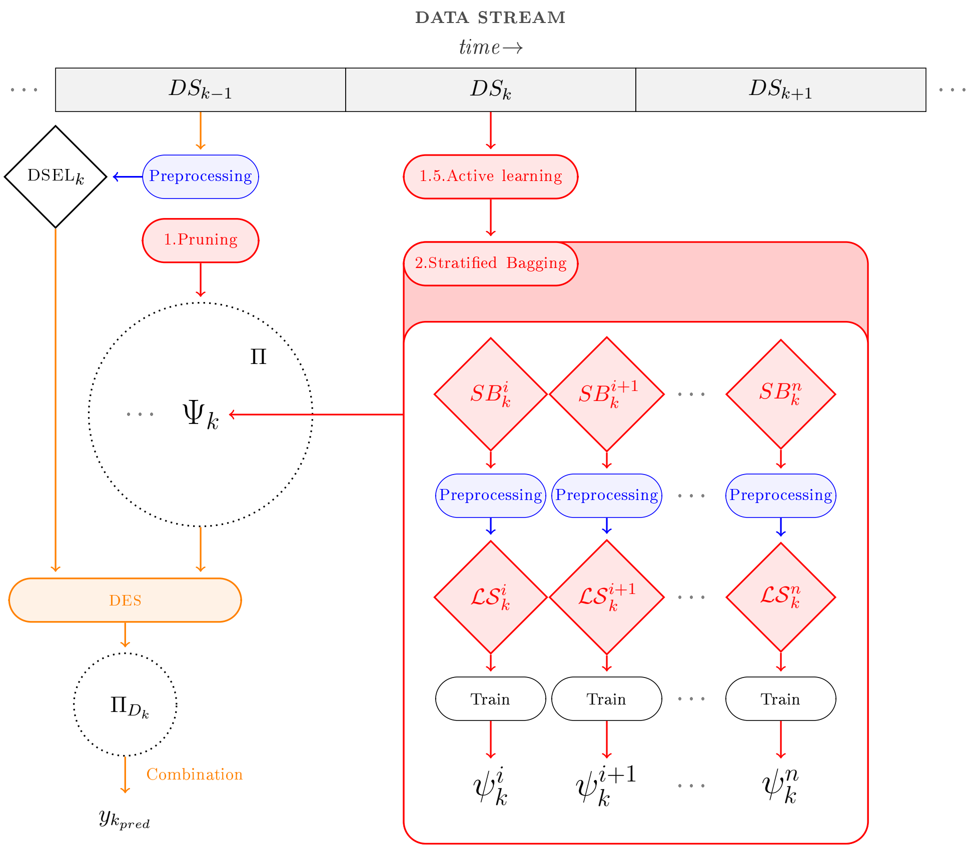


Figure 5.5: *The framework extended with the active learning module for training base classifiers and to prepare a* DSEL *for dynamic selection process. Here, T_k is the training data produced by preprocessing (Preproc) data chunk DS_k and Ψ_k is the base classifier trained on the kth data chunk. E denotes the classifier pool.*

The algorithms described in Section 5.1 will be reused as labeling methods. The first is Budget Labeling (BLS) which trains each new base classifier on a fixed percentage of randomly selected problem instances from the current chunk (Algorithm 15). The second method is the ALS that has been modified. As before, this algorithm selects patterns that are within a certain distance from the problem's decision boundary defined by the threshold t, but this time it can also be given the budget b, which defines the percentage of these patterns we want to label. ALS pseudocode is presented in Algorithm 16.

As the framework is supposed to work with highly imbalanced problems, another modification has been made to labeling methods. If all the labeled instances come from the same class, a new model is not added to the classifier pool.

Algorithm 15 Pseudocode for BLS

```
Input:
    $Stream = \{\mathcal{DS}_1, \mathcal{DS}_2, \ldots, \mathcal{DS}_k, \mathcal{DS}_{k+1}, \ldots\}$ – data stream,
    $\Psi$ – classification algorithm,
    $b$ – budget value.

1: for each $k, \mathcal{DS}_k = \{x_k^1, x_k^2, \ldots, x_k^N\}$ in $Stream$ do
2:     $\mathcal{X}_k$ = randomBudget($b, \mathcal{DS}_k$)      ▷ Randomly select percentage of instances
3:     $\mathcal{LS}_k$ = getLabels($\mathcal{X}_k$)      ▷ Get labels for the chosen instances
4:     $\Psi \leftarrow$ updateClassifier($\Psi, \mathcal{LS}_k$)      ▷ Update the classifier
5: end for
```

Algorithm 16 Pseudocode for the modified ALS

```
Input:
    $Stream = \{\mathcal{DS}_1, \mathcal{DS}_2, \ldots, \mathcal{DS}_k, \mathcal{DS}_{k+1}, \ldots\}$ – data stream,
    $\Psi$ – classification algorithm,
    $t$ – threshold,
    $b$ – budget.

1: for each $k, \mathcal{DS}_k = \{x_k^1, x_k^2, \ldots, x_k^N\}$ in $Stream$ do
2:     if $k == 0$ then
3:         $\Psi \leftarrow$ updateClassifier($\Psi, \mathcal{DS}_k$)  ▷ Update the classifier using whole chunk
4:     else
5:         $\mathcal{X}_k$ = activeLearning($t, b, \mathcal{DS}_k$)      ▷ Select instances using *active learning*
6:         $\mathcal{LS}_k$ = getLabels($\mathcal{X}_k$)
7:         $\Psi \leftarrow$ updateClassifier($\Psi, \mathcal{LS}_k$)      ▷ Update the classifier
8: end for
```

Computational and memory complexity analysis

The computational complexity of DESISC-SB framework is based on the *Dynamic Ensemble Selection* methods of as well as on preprocessing techniques. The key factors affecting the computational complexity of the presented approaches are, respectively, the number of models in the classifier pool for *Dynamic Selection* algorithms and the number of problem instances in a single data chunk in the case of preprocessing techniques.

Based on preliminary observations, it was established that the KNORA-U has a linear time complexity of $O(n)$ depending on the number of base classifiers in the pool. The ROS preprocessing technique has the logarithmic complexity of $O(log\ n$. Stratified Bagging performs sampling with replacement for each class with computational complexity of $O(\mid i \mid n)$, where $\mid i \mid$ is the cardinality of the ith class and n denotes the number of *bootstraps* (number of base models in bagging classifier) [79].

The BLS algorithm uses a simple sampling without replacement in order to choose the random budget b of instances from each data chunk $\mathcal{DK}_k$. This operation has the computational complexity of $O(b\ log\ b)$. The used *active learning* approach calculates each sample's distance from the decision boundary (which an absolute difference of obtained support and .5), which has the complexity of $O(\mid \mathcal{DS}_k \mid)$. Then, ALS sorts the objects according to the acquired distance and uses only those, for which the distance values does not exceed the set threshold t. This operation has the computational complexity of $O(\mid \mathcal{DS}_k \mid log \mid \mathcal{DS}_k \mid)$.

5.2.1 Experimental evaluation

Here, the motivation, goals and set-up of the performed experiments are presented.

Research questions

The experiments were designed to answer the following questions:

Q1. Is the batch-based DESISC-SB framework for imbalanced data stream classification, introduced in Section 4.2, compatible with active learning methods?

Q2. Is it possible to control the metric values obtained in the task of imbalanced data stream classification by parametrization of the threshold t in the ALS method?

Goals of the experiments

Experiment 1 – The impact of active learning on the DSEISC-SB *framework*

The aim of the first experiment is to see how the use of a data labeling strategy affects the results achieved by the proposed framework.

Experiment 2 – The impact of the ALS *distance threshold on the values of evaluation metrics*

The aim of the second experiment is to check whether the obtained metric values can be controlled by changing the distance from the decision boundary on the basis of which the ALS selects patterns for labeling.

Experimental set-up

The analysis was based on six types of synthetic streams, replicated 10 times for stability of the achieved results. The detailed characteristics of the generated streams are described below:

- *Concept drift* types – *sudden*, *gradual* and *incremental*,
- Approaches to repetitive concepts – *recurrent* and *non-recurrent concept drift*,
- Data stream size – 50 000 instances (200 data chunks, 250 instances each)
- Number of concept drift per stream – 9,
- Global label noise – 5%,
- Imbalance Ratio – 19.

Additionally, experiments were carried out on 5 real data streams, the characteristics of which are presented in the table 5.2.

Table 5.2: *Real data streams characteristics.*

Data stream	#Samples	#Features	IR
covtypeNorm-1-2vsAll	266 000	54	4
poker-lsn-1-2vsAll	360 000	10	10
INSECTS-abrupt_imbalanced_norm	300 000	33	19
INSECTS-gradual_imbalanced_norm	100 000	33	19
INSECTS-incremental_imbalanced_norm	380 000	33	19

The experimental evaluation was carried out in accordance with the *Test-Then-Train* methodology. The DESISC-SB framework presented in section 4.2.2 was chosen as the classifier. Its parameters (i.e. dynamic selection method and preprocessing technique) were selected based on the results of the experiments performed in section 4.2.3.2 and are listed below:

- Base classifier – Naïve Bayes Classifier,
- *Dynamic Ensemble Selection* – KNORA-U at the level of bagging classifiers,
- Data preprocessing – *Random Oversampling*,
- Fixed classifier pool size – 5 bagging classifiers, 10 base classifiers each (50 models in total).

Comparative methods:

- Whole - model updated using all available data,
- BLS-15 - 15% of random budget,
- ALS-15 - 15% of instances closes to the decision boundary,
- ALS - all instances within distance of 0.2 from the decision boundary.

The methods' parameters were selected based on the experience gained during research on the BALS algorithm and also taking into account the batch approach and base classifier used.

Experiment 1 – The impact of active learning on the DSEISC-SB framework
Figure 5.6 shows the results of using the proposed framework in the case of sudden concept drifts occurrence. The first thing that stands out is that the BLS-15 result is similar to that of the random classifier. This is due to the high imbalance ratio (5% of minority class) in the data stream. Because of that BLS selects only instances belonging to the majority class and the new model is not added to the pool. At the same time, we can see that both ALS-15 and ALS are doing relatively well. Both in the case of recurring and non-recurring drift, ALS is better at identifying the minority class, due to the lack of a set budget. Thanks to this, it maintains a high generalization capacity and in some cases is able to perform similarly to the model learned on all available data.

Figure 5.7 shows the results obtained in the case of gradual drift, characterized by slower dynamics of change and the occurrence of instances from both concepts at the same time. In this case, for non-recurring drift, we can observe a progressive deterioration of the generalizing ability of ALS. This may be due to a small number of instances located within a given distance from the decision boundary, which in turn leads to underfitting in the face of a constant concept change. On the other hand, in the case of recurring gradual drift, the ALS and ALS-15 remain on a similar level, because the ensemble always includes models that remember the old concept.

In the case of the of incremental drift occurrence (Figure 5.8) the observations are similar to those regarding the gradual concept drift. The difference is that whether the drift is recurring or non-recurring, the ALS-15 and ALS methods achieve nearly identical results. This may be the result of more instances available to ALS as one concept blends seamlessly into another.

Figure 5.9 shows the results of combining the DESISC-SB framework with active learning methods in a classification task of five real imbalanced data streams. Radar charts show

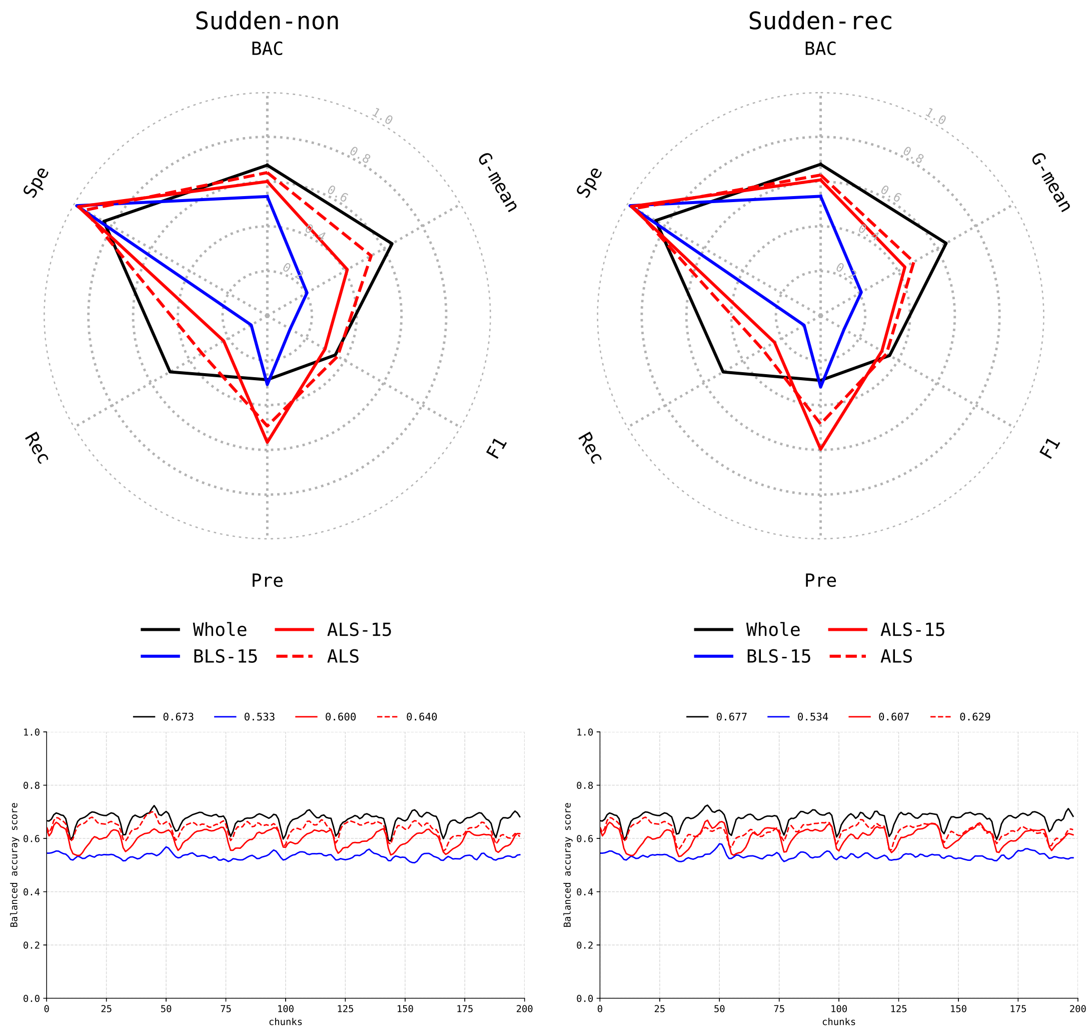


Figure 5.6: *Results for sudden drift.*

values of six metrics averaged over the entire length of the stream, while the runs are shown for the $Gmean_s$ metric. Due to the use of the batch-based data stream classification approach and the GNB classifier as the base model, it was decided to abandon the BLS approach, which in this case would remain at the level of the random classifier.

In the case of the CovType stream, the ALS-15 approach - selecting 15% of the instances closest to the problem's decision boundary - achieves a generalization ability worse than the full model. At the same time, however, the selection based on the distance to decision boundary allows for the selection of instances belonging to both classes for later data preprocessing, and the lower performance is a direct result of the smaller training set size. Interestingly, the model learning with the use of ALS almost immediately drops to the level of a random classifier and stays there along the entire length of the stream. This may be due to the support space distribution, in which, due to the high certainty of

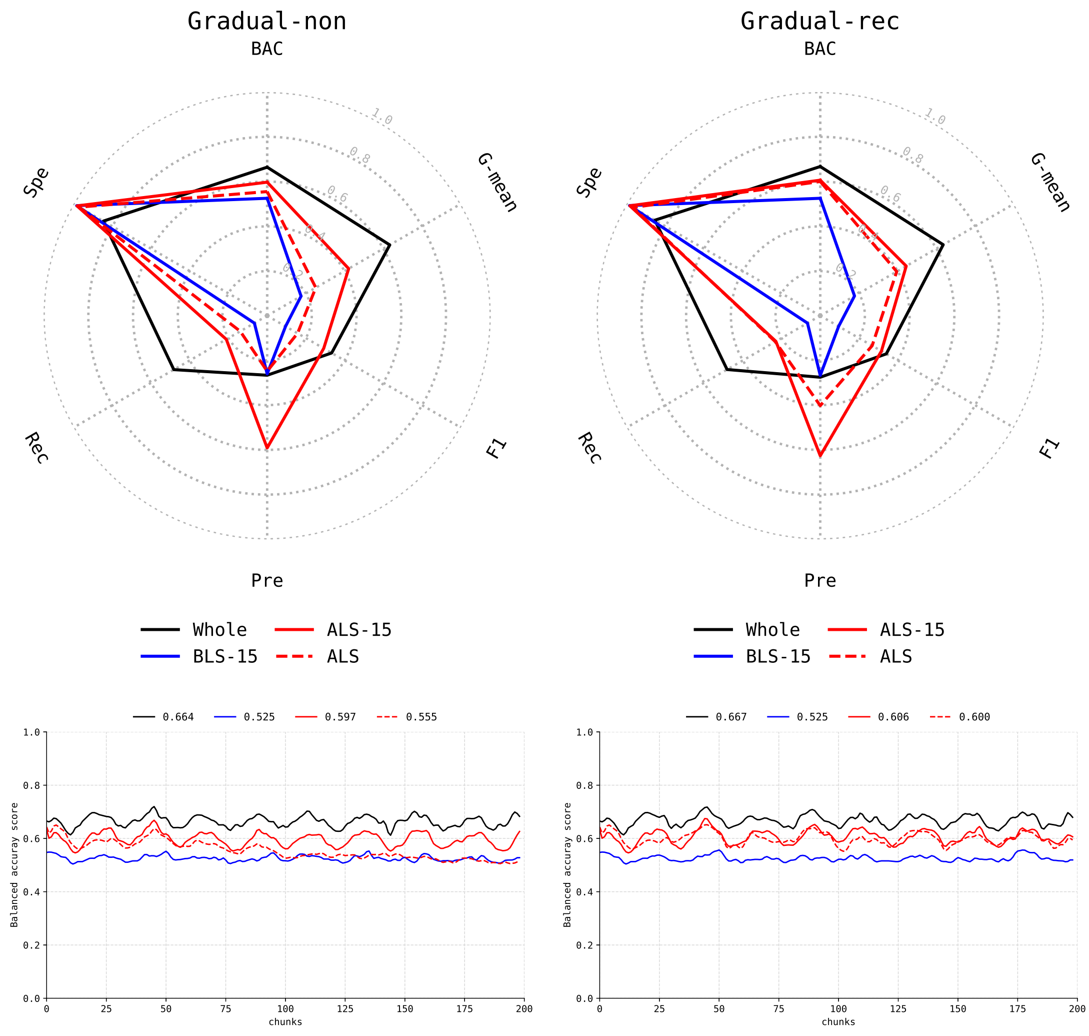


Figure 5.7: *Results for gradual drift.*

the model used, there are no instances lying within a defined distance from the decision boundary.

For the slightly more difficult Poker stream, both the ALS and ALS-15 methods show similar behavior. ALS-15 can achieve generalization capacity close to the full model but also shows greater model degradation when concept drifts occur. The ALS approach is more stable, which is due to the collection of more training patterns in the event of concept drift, as the labeling limitation with this method does not concern the number of patterns, but only the distance to the decision boundary.

The observations related to the classification of INSECTS streams, presenting three defined types of concept drift, are particularly interesting. In the case of sudden drift, the model learned using the ALS approach achieves the generalization ability at the level of the full model. It may be caused by low classification certainty, and thus a large number of patterns located at a given distance from the decision boundary. The model using

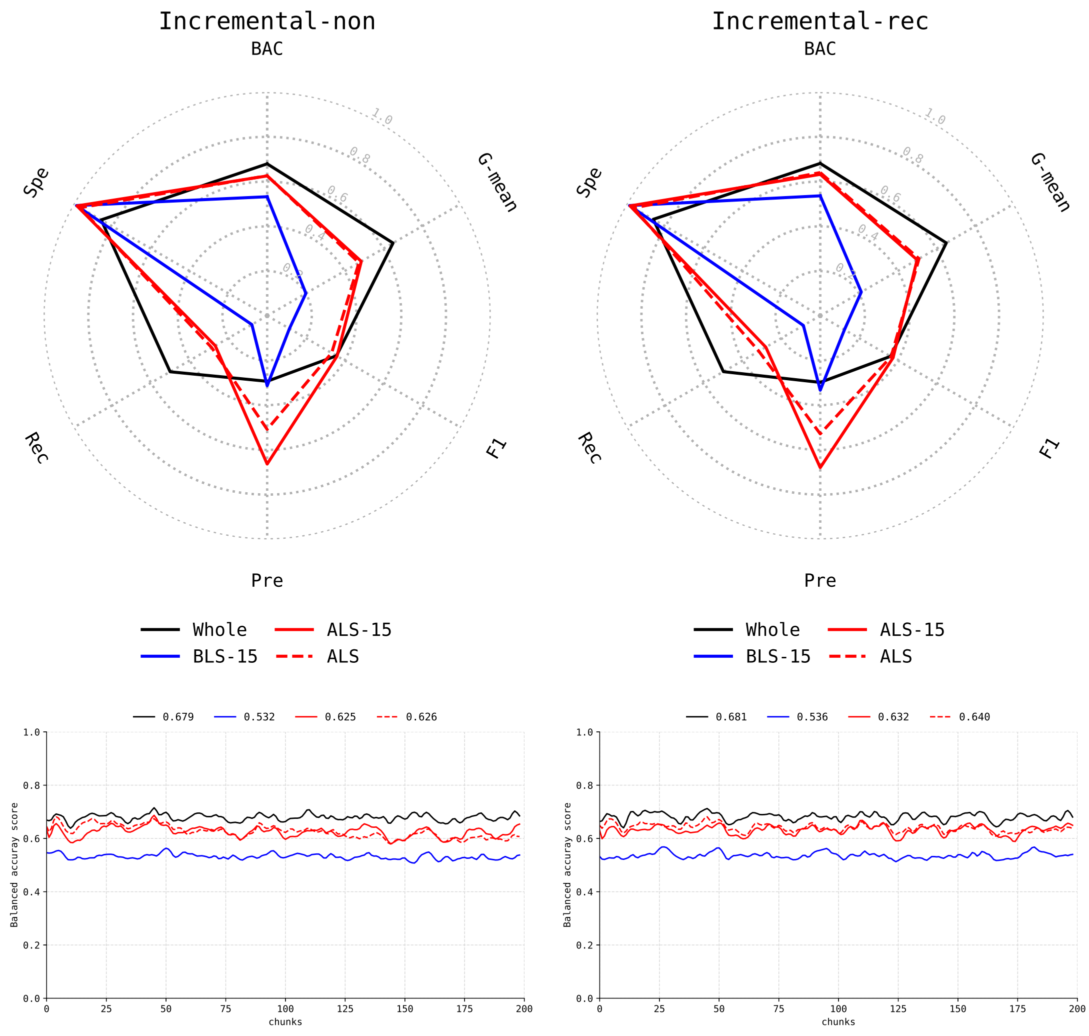


Figure 5.8: *Results for incremental drift.*

the ALS-15 approach achieves slightly lower results than ALS, which is a direct result of the smaller number of training patterns available. In the case of gradual drift, all three approaches have very similar performance. This is due to the drift characteristics and proves that only a small number of instances closest to the decision limit is sufficient for building a useful model. When dealing with incremental concept drift, the model learned using the ALS-15 approach displays a correspondingly lower generalization capacity, resulting from the smaller number of patterns used for updating the classifier. At the same time, however, this model is relatively stable compared to the classifier trained using the ALS method, which demonstrates greater degeneration in the event of concept drift occurrence. This is due to the changes in the support space and the lack of patterns that can be used during the training process.

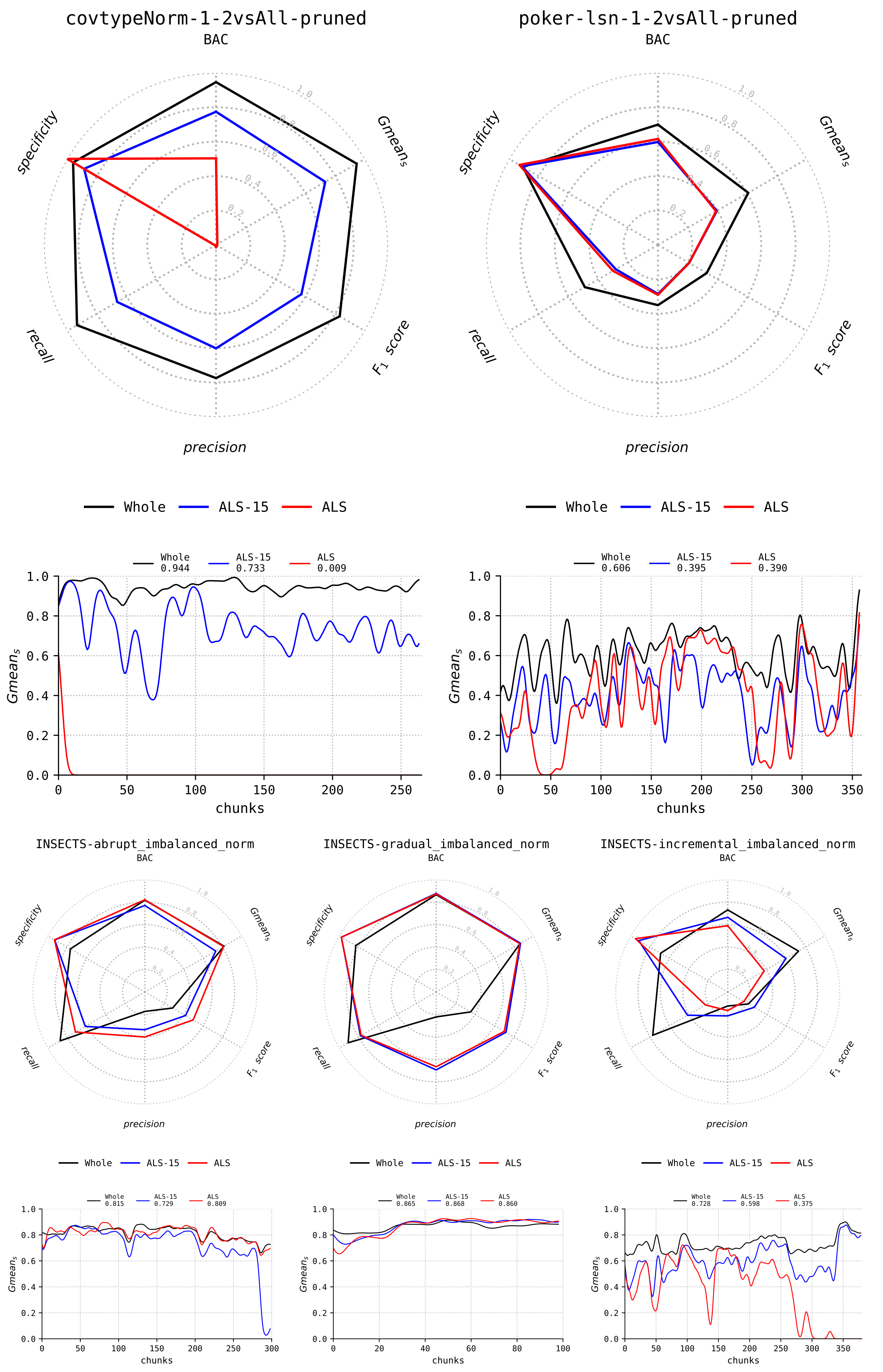


Figure 5.9: *Results of the MDE comparison with reference methods for real data streams.*

Experiment 2 – The impact of the ALS distance threshold on the values of evaluation metrics

Figure 5.10 shows the averaged results of the evaluation metrics for each type of concept drift, depending on the value of threshold t. Additionally, the X-axis shows the average percentage of instances used in the training process, and the right Y-axis shows the metric values achieved by the model trained on the entire available data.

Regardless of the type of drift and whether it is recurring or non-recurring, we can observe more or less the same dependencies for each of the evaluation metrics. The value of specificity, which is responsible for the ability to recognize the majority class, decreases with an increase in the number of used samples, which in turn causes an increase in the value of recall. This is a typical phenomenon in the problem of imbalanced data classification as the two metrics are closely related.

Especially interesting is the behavior of precision metric, which increases until the value of t is approximately .20 or .25 and then starts to decline. This is a sign that model started to prefer the minority class.

The values of aggregated metrics, i.e. BAC, $Gmean_s$, and F_1 *score*, result directly from the values of the base metrics. Balanced Accuracy score and $Gmean_s$ note a continuous increase that slows significantly when t achieves the value of 0.25 or 0.3. At the same time, the F1 score usually reaches its highest value due to the significant increase in precision.

Observations

Based on the results obtained, it can be concluded that the proposed batch-based framework for imbalanced data stream classification is compatible with active learning methods. ALS works especially well in the case of sudden drift, where about 25% of the instances closest to the decision boundary are sufficient to achieve results similar to the model trained using all instances of the problem.

Research on real data streams has shown that the ALS approach - using all patterns within a given distance from the problem's decision boundary - cannot be used in its current form for every data stream. This is due to the high sensitivity of the method to the distribution of patterns in the support space, which, if the classification is too certain, leads to the lack of patterns that can be used in the model training process. To deal with this problem, threshold t should not be set as a fixed parameter, but rather optimized for each data chunk, to ensure that models using this approach always get a training set containing patterns useful in the training process.

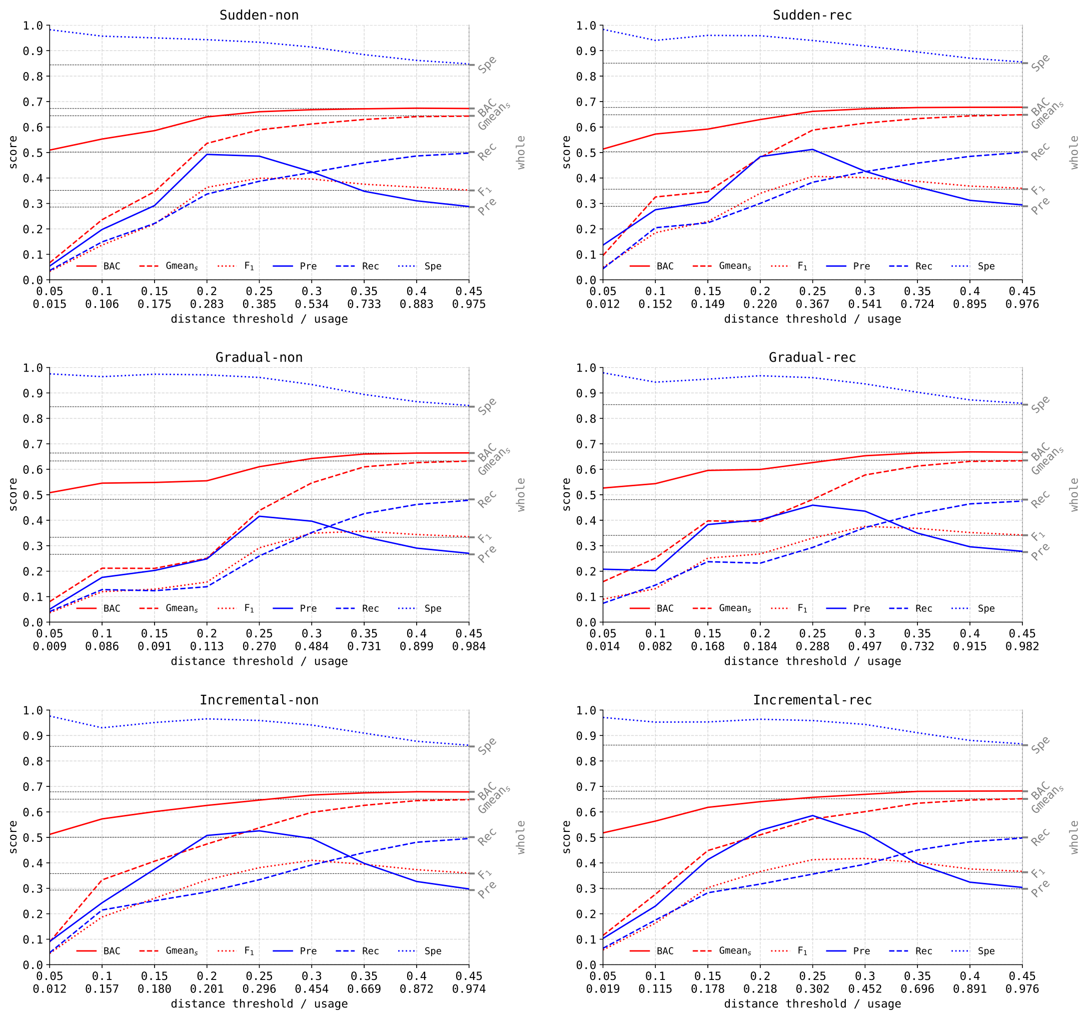


Figure 5.10: *Average metric values in relation to the set threshold t.*

Answers to research questions

Q1. Is the batch-based DESISC-SB framework for imbalanced data stream classification, introduced in Section 4.2, compatible with active learning methods?

A1. Obtained results confirmed, that the DESISC-SB batch-based framework for imbalanced data stream classification is compatible with active learning approaches.

Q2. Is it possible to control the metric values obtained in the task of imbalanced data stream classification by parametrization of the threshold t in the ALS method?

A2. Based on the conducted experiments, it can be concluded that by appropriate parametrization of the ALS method, it is possible to optimize the *precision* metric.

Chapter 6

Conclusions and Future Works

This thesis focused on the use of dynamic ensemble selection methods and data preprocessing techniques in the problem of streaming and imbalanced data classification. This work showed the potential of classifier selection methods to deal with class imbalance, but also, most of all, proposed new effective solutions to the problem of highly imbalanced data stream classification, up to this point rarely discussed in the literature. The stated hypothesis – that *there exist such methods employing data preprocessing and classifier selection that can outperform state-of-the-art classifiers for difficult data classification tasks* – seems to be proven by achieving the following goals:

1. **Developing an ensemble selection algorithm for imbalanced data classification, as well as designing a dedicated combination rule.**
This goal was met by developing three algorithms based on the clustering of models in a one-dimensional space of classifier diversity. The clustering space was based on the proposed H measure, which informs about the impact of individual classifiers on the diversity achieved by the entire ensemble.

The *Diversity Ensemble Pruning* (DEP) algorithm groups the base models in the diversity space and then evaluates them in terms of *balanced accuracy*. The pruned ensemble consists of the classifiers with the highest BAC in each cluster. The *Two-step majority voting organization* (TSMV) algorithm classifies imbalanced data using the two-step voting structure, instead of pruning the ensemble. In the first stage of voting, each cluster is treated as a separate classifier pool, which independently makes a decision based on the *majority voting*. In the second step, the *majority voting* procedure is repeated, combining the decisions obtained by the individual clusters. The *Random Sampling Multistage Organization* (RSMO) algorithm, which is a modification of TSVM, additionally uses sampling with replacement to reduce the number of similar classifiers used in the decision-making process.

The computer experiments on imbalanced data, as well as statistical analysis confirmed the usefulness of the proposed pruning method and showing it's potential for increasing the ensemble's ability do detect the minority class.

The first proposition of ensemble methods based on clustering in diversity space was published in [281]. The proposals were then extended in [283] and evaluated for the imbalanced data classification in [276].

2. Proposing a novel distance-based *Dynamic Ensemble Selection* method for imbalanced data classification.
This goal has been achieved by proposing a novel solution based on dynamic classifier selection for imbalanced data classification problem. Two methods were proposed, namely DESE and DESIRE, which use the Euclidean distance and *Imbalance Ratio* in the training set to select the most appropriate model for the classification of each new sample. Research conducted on benchmark datasets and statistical analysis confirmed the usefulness of proposed methods, especially when there is a need to maintain a relatively low number of classifiers.

The propositions of DESE and DESIRE were first published in [282].

3. Developing a chunk-based ensemble algorithm, aimed specifically for the task of highly imbalanced data stream classification.
This goal was met by proposing a novel, *Minority Driven Ensemble* method for a challenging task of imbalanced data stream classification. MDE employs dynamic classifier selection approach to exploit local data characteristics. The computer experiments confirmed the usefulness of the proposed method and on the basis of a thorough statistical analysis.

The proposition of MDE was first published in [277].

4. Designing a novel framework combining *Dynamic Ensemble Selection* and preprocessing techniques for imbalanced data stream classification.
A novel DESISC framework combining *Dynamic Ensemble Selection* and preprocessing techniques (both *oversampling* and *undersampling*) was proposed for the task of highly imbalanced data streams. The extended version of this approach, named DESISC-SB, is based on using bagging classifiers diversified using *stratified bagging*, which performs sampling with replacement separately from the minority and majority class. The research

conducted on two dynamic selection methods (in two variants) and four preprocessing techniques confirmed the effectiveness of the proposed solution and highlighted its strengths in comparison with *state-of-art* methods. The proposed framework, compared to the reference methods, was characterized by balanced performance in terms of all evaluation metrics which was stable regardless of the imbalance ratio or concept drift type. Thus, the validity of using the *Dynamic Classifier Selection* methods to classify drifting imbalanced data streams was confirmed. The obtained results are showing the way for further research on employing local classifier competences for difficult data stream classification.

The first proposition of DESISC framework was published in [280]. The extended DESISC-SB framework was proposed in [284].

5. Proposing a strategy for learning from drifting data stream under limited access to labels scenario.
The modification of the active learning method dedicated to non-stationary data stream classifiers was introduced. The proposed BALS algorithm, in addition to the pool of objects selected for labeling (according to the rule that objects close to decision boundaries have a large impact on model modification), also received a small number of randomly selected objects from among the other instances belonging to an analyzed data chunk. This approach caused the classifier to stabilize faster after the *concept drift* than BLS or ALS. Also, the deterioration of BALS quality is lower than the reference algorithms.

The proposition of BALS was first published in [278].

6. Evaluating the behavior of the previously proposed data stream classification framework, taking into account the limitation in the label access.
This goal was achieved by combining the proposed framework with the active learning method based on selecting patterns located at a certain distance from the decision boundary. The conducted research confirmed the usefulness of the framework under a high imbalance ratio and limited access to labels.

7. Conducting an experimental evaluation of the proposed methods in comparison to *state-of-the-art* approaches.
8. Developing a *Python* Machine Learning library for difficult data stream analysis.
These goals were achieved by designing an experimental environment for static classification problems (For Chapter 3), as well as by designing the *stream-learn* package for

difficult data stream classification which was used to conduct all experiments presented in Chapters 4 and 5.

The *stream-learn* package has already been tested in the research process of preparing several scientific articles, and it is an ideal tool for users who care about the simplicity of processing, ease of the use and integration with the *scikit-learn* machine learning library. The article describing package contents is available on *arXiv* [141].

Future works

The ideas presented in this thesis may be potentially developed in the following directions:

- Future research on clustering-based methods for ensemble pruning may include exploring the different ways of calculating the proposed H measure (including both deterministic and non-deterministic variants) and, in case of multistage organization methods, employing different types of voting (e.g. weighted majority voting). It would be useful to also consider ways of dealing with ties during the voting process and, possibly, investigate the effects of data dimensionality on the performance of the proposed algorithms.
- Future work on distance-based DES may involve the exploration of different approaches to the base classifiers' weighting, as well as using different combination methods and the use of proposed methods for the imbalanced data stream classification.
- The MDE algorithm can be extended for other types of base classifiers, e.g. by taking into account the threshold for the minority class supports returned by each of the models in the ensemble.
- Further research regarding DESISC and DESISC-SB frameworks for imbalanced data stream classification may include problems with multiple concept drifts. A comprehensive analysis employing measures used to evaluate the behavior of methods during concept drift occurrence, their extension for many concept drifts in a single the data stream, and their statistical analysis. It is also possible to extend the research to other methods of *Dynamic Ensemble Selection* and preprocessing techniques, as well as to adapt our proposition to the multi-class classification task.
- Employing the BALS method for another classification models and classifier ensembles, as well as using the information on the *concept drift* rapidness to establish the proportion between the number of objects labeled by active learning and random choosing.

- Conducting a broader experimental evaluation of the proposed framework for the imbalance data stream classification under a scenario of restricted access to labels.

Publications

The selected parts of the thesis have been already published in:

- Paweł Zyblewski, Robert Sabourin, and Michał Woźniak. Preprocessed dynamic classifier ensemble selection for highly imbalanced drifted data streams. *Information Fusion*, 66:138 – 154, 2021 (**IF: 13.669, MNiSW: 200**)
- Paweł Zyblewski and Michał Woźniak. Novel clustering-based pruning algorithms. *Pattern Analysis and Applications*, pages 1–10, 2020 (**IF: 1.512, MNiSW: 70**)
- Paweł Zyblewski, Robert Sabourin, and Michał Woźniak. Data preprocessing and dynamic ensemble selection for imbalanced data stream classification. In *Machine Learning and Knowledge Discovery in Databases*, pages 367–379, Cham, 2020. Springer International Publishing (**CORE: A, MNiSW: 140**)
- Paweł Zyblewski and Michał Woźniak. Dynamic classifier selection for data with skewed class distribution using imbalance ratio and euclidean distance. In *International Conference on Computational Science*, pages 59–73. Springer, 2020 (**CORE: A, MNiSW: 140**)
- Paweł Zyblewski. Clustering-based ensemble pruning in the imbalanced data classification. In *International Conference on Computational Science*. Springer, 2021 [accepted for publication] (**CORE: A, MNiSW: 140**)
- Paweł Zyblewski, Paweł Ksieniewicz, and Michał Woźniak. Classifier selection for highly imbalanced data streams with minority driven ensemble. In *Artificial Intelligence and Soft Computing*, pages 626–635, Cham, 2019. Springer International Publishing (**CORE: National, MNiSW: 20**)
- Paweł Zyblewski, Paweł Ksieniewicz, and Michał Woźniak. Combination of active and random labeling strategy in the non-stationary data stream classification. In *International Conference on Artificial Intelligence and Soft Computing*, pages 576–585. Springer, 2020 (**CORE: National, MNiSW: 20**)
- Paweł Zyblewski and Michał Woźniak. Clustering-based ensemble pruning and multistage organization using diversity. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 287–298. Springer, 2019 (**CORE: National, MNiSW: 20**)

- Paweł Ksieniewicz and Paweł Zyblewski. stream-learn–open-source python library for difficult data stream batch analysis. *arXiv preprint arXiv:2001.11077*, 2020

During the work on the thesis I have also coauthored other research:

- P. Ksieniewicz, P. Zyblewski, M. Choraś, R. Kozik, A. Giełczyk, and M. Woźniak. Fake news detection from data streams. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020 (**CORE: A, MNiSW:** 140)
- Dominika Sułot, Paweł Zyblewski, and Paweł Ksieniewicz. Analysis of variance application in the construction of classifier ensemble based on optimal feature subset for the task of supporting glaucoma diagnosis. In *International Conference on Computational Science*. Springer, 2021 [accepted for publication] (**CORE: A, MNiSW:** 140)
- Paweł Zyblewski, Marek Pawlicki, Rafał Kozik, and Michał Choras. Cyber-attack detection from iot benchmark considered as data streams. In *International Conference on Computer Recognition Systems*, 2021 [accepted for publication]

Additionally, at the time of completing this thesis, the following articles are undergoing the review process:

- Paweł Ksieniewicz, Paweł Zyblewski, and Robert Burduk. Fusion of linear base classifiers in geometric space. *Knowledge-Based Systems*, 2021
- Paweł Ksieniewicz, Paweł Zyblewski, Weronika Borek, Rafał Kozik, Michał Choras, and Michał Wozniak. Alphabet flatting as a variant of n-gram feature extraction method in ensemble classification of fake news. *Engineering Applications of Artificial Intelligence*, 2021
- Paweł Ksieniewicz and Paweł Zyblewski. stream-learn–open-source python library for difficult data stream batch analysis. *Neurocomputing*, 2021
- Joanna Komorniczak, Paweł Zyblewski, and Paweł Ksieniewicz. Prior probability estimation in dynamically imbalanced data streams. In *The International Joint Conference on Neural Networks*, 2021

The research presented in this thesis was supported by the Polish National Science Centre under the grant No. 2017/27/B/ST6/01325.

Bibliography

[1] Zahraa Said Abdallah, Mohamed Medhat Gaber, Bala Srinivasan, and Shonali Krishnaswamy. Adaptive mobile activity recognition system with evolving data streams. *Neurocomputing*, 150:304–317, 2015.

[2] Tamer S Abdelgayed, Walid G Morsi, and Tarlochan S Sidhu. Fault detection and classification based on co-training of semisupervised machine learning. *IEEE Transactions on Industrial Electronics*, 65(2):1595–1605, 2017.

[3] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional space. In Jan Van den Bussche and Victor Vianu, editors, *Database Theory — ICDT 2001*, pages 420–434, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[4] Jesus Alcala-Fdez, Alberto Fernández, Julián Luengo, J. Derrac, S Garc'ia, Luciano Sanchez, and Francisco Herrera. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17:255–287, 01 2010.

[5] Ethem Alpaydin. *Introduction to Machine Learning, Fourth Edition.* MIT press, 2020.

[6] Mohamed Aly. Survey on multiclass classification methods. *Neural Netw*, 19:1–9, 2005.

[7] Dana Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.

[8] Les E Atlas, David A Cohn, and Richard E Ladner. Training connectionist networks with queries and selective sampling. In *Advances in neural information processing systems*, pages 566–573. Citeseer, 1990.

[9] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[10] B. Bakker and T. Heskes. Clustering ensembles of neural network models. *Neural Networks*, 16(2):261–269, 2003.

[11] Ricardo Barandela, Josep Sánchez, Vicente García, and E. Rangel. Strategies for learning in class imbalance problems. *Pattern Recognition*, 36:849–851, 03 2003.

[12] Saman Bashbaghi, Eric Granger, Robert Sabourin, and Guillaume-Alexandre Bilodeau. Robust watch-list screening using dynamic ensembles of svms based on multiple face representations. *Machine vision and applications*, 28(1-2):219–241, 2017.

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

[14] Eric B Baum and Kenneth Lang. Query learning can work poorly when a human oracle is used. In *International joint conference on neural networks*, volume 8, page 8, 1992.

[15] FRS Bayes. An essay towards solving a problem in the doctrine of chances. *Biometrika*, 45(3-4):296–315, 1958.

[16] Simon Bernard, Clément Chatelain, Sébastien Adam, and Robert Sabourin. The multiclass roc front method for cost-sensitive classification. *Pattern Recognition*, 52:46 – 60, 2016.

[17] U. Bhowan, M. Johnston, M. Zhang, and X. Yao. Evolving diverse ensembles using genetic programming for classification with unbalanced data. *IEEE Transactions on Evolutionary Computation*, 17(3):368–386, June 2013.

[18] 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.

[19] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. MOA: massive online analysis. *J. Mach. Learn. Res.*, 11:1601–1604, 2010.

[20] Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Leveraging bagging for evolving data streams. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 135–150. Springer, 2010.

[21] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, and Ricard Gavalda. Improving adaptive bagging methods for evolving data streams. In *Asian conference on machine learning*, pages 23–37. Springer, 2009.

[22] Christopher M Bishop. *Pattern recognition and machine learning.* springer, 2006.

[23] Rico Blaser and Piotr Fryzlewicz. Random rotation ensembles. *The Journal of Machine Learning Research*, 17(1):126–151, 2016.

[24] Jerzy Błaszczyński, Jerzy Stefanowski, and Roman Słowiński. Consistency driven feature subspace aggregating for ordinal classification. In Víctor Flores, Fernando Gomide, Andrzej Janusz, Claudio Meneses, Duoqian Miao, Georg Peters, Dominik Ślęzak, Guoyin Wang, Richard Weber, and Yiyu Yao, editors, *Rough Sets*, pages 580–589, Cham, 2016. Springer International Publishing.

[25] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

[26] Dibya Jyoti Bora, Dr Gupta, and Anil Kumar. A comparative study between fuzzy clustering algorithm and hard clustering algorithm. *arXiv preprint arXiv:1404.6059*, 2014.

[27] Remco R Bouckaert. Estimating replicability of classifier learning experiments. In *Proceedings of the twenty-first international conference on Machine learning*, page 15, 2004.

[28] Mohamed-Rafik Bouguelia, Yolande Belaïd, and Abdel Belaïd. An adaptive streaming active learning strategy based on instance weighting. *Pattern Recognition Letters*, 70:38–44, 2016.

[29] Paula Branco, Luís Torgo, and Rita P Ribeiro. A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys (CSUR)*, 49(2):1–50, 2016.

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

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

[32] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees.* CRC press, 1984.

[33] Alceu S Britto Jr, Robert Sabourin, and Luiz ES Oliveira. Dynamic selection of classifiers—a comprehensive review. *Pattern recognition*, 47(11):3665–3680, 2014.

[34] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M. Buhmann. The balanced accuracy and its posterior distribution. In *Proceedings of the 2010 20th International Conference on Pattern Recognition*, ICPR '10, pages 3121–3124, Washington, DC, USA, 2010. IEEE Computer Society.

[35] Robert Bryll, Ricardo Gutierrez-Osuna, and Francis Quek. Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. *Pattern recognition*, 36(6):1291–1302, 2003.

[36] D. Brzezinski, J. Stefanowski, R. Susmaga, and I. Szczęch. On the dynamics of classification measures for imbalanced and streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–11, 2019.

[37] 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.

[38] Dariusz Brzeziński and Jerzy Stefanowski. Accuracy updated ensemble for data streams with concept drift. In Emilio Corchado, Marek Kurzyński, and Michał Woźniak, editors, *Hybrid Artificial Intelligent Systems*, pages 155–163, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

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

[40] Dariusz Brzezinski and Jerzy Stefanowski. Prequential auc for classifier evaluation and drift detection in evolving data streams. In *International Workshop on New Frontiers in Mining Complex Patterns*, pages 87–101. Springer, 2014.

[41] 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.

[42] 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.

[43] C. Bunkhumpornpat, K. Sinapiromsaran, and C. 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, 13th Pacific-Asia Conference 2009, Bangkok, Thailand, April 27-30, 2009, Proceedings*, pages 475–482, 2009.

[44] Robert Burduk. The adaboost algorithm with linear modification of the weights. In Michał Choraś and Ryszard S. Choraś, editors, *Image Processing and Communications Challenges 9*, pages 82–87, Cham, 2018. Springer International Publishing.

[45] Robert Burduk, Wojciech Bożejko, and Szymon Zacher. Novel approach to gentle adaboost algorithm with linear weak classifiers. In Ngoc Thanh Nguyen, Kietikul Jearanaitanakij, Ali Selamat, Bogdan Trawiński, and Suphamit Chittayasothorn, editors, *Intelligent Information and Database Systems*, pages 600–611, Cham, 2020. Springer International Publishing.

[46] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

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

[48] Marie Jean AN Caritat. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix.* De l'Imprimerie royale, 1785.

[49] 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.

[50] 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.

[51] Dong Chen, Xiao-Jun Wang, and Bin Wang. A dynamic decision-making method based on ensemble methods for complex unbalanced data. In *Web Information Systems Engineering – WISE 2019*, pages 359–372, Cham, 2019. Springer International Publishing.

[52] S. Chen, H. He, and E. A. Garcia. RAMOBoost: Ranked minority oversampling in boosting. *IEEE Transactions on Neural Networks*, 21(10):1624–1642, 2010.

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

[54] Xue-wen Chen and Michael Wasikowski. Fast: A ROC-based feature selection metric for small samples and imbalanced data classification problems. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 124–132, 2008.

[55] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994.

[56] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[57] Rafael M. O. Cruz, Luiz G. Hafemann, Robert Sabourin, and George D. C. Cavalcanti. DESlib: A Dynamic ensemble selection library in Python. *arXiv preprint arXiv:1802.04967*, 2018.

[58] Rafael M. O. Cruz, Robert Sabourin, and George D. C. Cavalcanti. Meta-des.oracle: Meta-learning and feature selection for dynamic ensemble selection. *Information Fusion*, 38:84–103, 2017.

[59] Rafael M. O. Cruz, Robert Sabourin, and George D. C. Cavalcanti. Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, 41:195–216, 2018.

[60] Rafael M.O. Cruz, Robert Sabourin, George D.C. Cavalcanti, and Tsang Ing Ren. Meta-des: A dynamic ensemble selection framework using meta-learning. *Pattern Recognition*, 48(5):1925–1935, 2015.

[61] Pádraig Cunningham and John Carney. Diversity versus quality in classification ensembles based on feature selection. In Ramon López de Mántaras and Enric Plaza, editors, *Machine Learning: ECML 2000*, pages 109–116, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[62] Jack Cuzick. A Wilcoxon-type test for trend. *Statistics in medicine*, 4(1):87–90, 1985.

[63] V. M. A. d. Souza, D. F. Silva, and G. E. A. P. A. Batista. Classification of data streams applied to insect recognition: Initial results. In *2013 Brazilian Conference on Intelligent Systems*, pages 76–81, 2013.

[64] Ido Dagan and Sean P Engelson. Committee-based sampling for training probabilistic classifiers. In *Machine Learning Proceedings 1995*, pages 150–157. Elsevier, 1995.

[65] Qun Dai. A competitive ensemble pruning approach based on cross-validation technique. *Know.-Based Syst.*, 37:394–414, January 2013.

[66] Belur V Dasarathy and Belur V Sheela. A composite classifier system design: Concepts and methodology. *Proceedings of the IEEE*, 67(5):708–713, 1979.

[67] Luc Devroye, László Györfi, and Gábor Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 2013.

[68] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[69] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, Aug 2000.

[70] Thomas G Dietterich and Ghulum Bakiri. Error-correcting output codes: A general method for improving multiclass inductive learning programs. In *AAAI*, pages 572–577. Citeseer, 1991.

[71] Thomas G Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286, 1994.

[72] G. Ditzler and R. Polikar. Incremental learning of concept drift from streaming imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 25(10):2283–2301, Oct 2013.

[73] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80, 2000.

[74] C. Dubos, S. Bernard, S. Adam, and R. Sabourin. Roc-based cost-sensitive classification with a reject option. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3320–3325, 2016.

[75] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification.* John Wiley & Sons, 2012.

[76] R. P. W. Duin. The combining classifier: to train or not to train? In *Object recognition supported by user interaction for service robots*, volume 2, pages 765–770 vol.2, Aug 2002.

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

[78] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[79] E. Fersini, E. Messina, and F.A. Pozzi. Sentiment analysis: Bayesian ensemble learning. *Decision Support Systems*, 68:26–38, 2014.

[80] Joseph L. Fleiss. *Statistical Methods for Rates and Proportions.* John Wiley & Sons, 1981.

[81] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and computation*, 121(2):256–285, 1995.

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

[83] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.

[84] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *Annals of statistics*, 28(2):337–407, 2000.

[85] Q. Fu, S. X. HU, and S.Y. Zhao. Clustering-based selective neural network ensemble. *Journal of Zhejiang University SCIENCE*, 6(5):387–392, 2005.

[86] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. 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, July 2012.

[87] Francis Galton. Vox populi, 1907.

[88] Joao Gama. *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC, 1st edition, 2010.

[89] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, Mar 2013.

[90] 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):37–49, 2008.

[91] Jing Gao, Philip S. Yu, Wei Fan, Bolin Ding, and Jiawei Han. Classifying data streams with skewed class distributions and concept drifts. *IEEE Internet Computing*, 12:37–49, 2008.

[92] Nicolas Garcia-Pedrajas and Domingo Ortiz-Boyer. Improving multiclass pattern recognition by the combination of two strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6):1001–1006, 2006.

[93] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

[94] 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.

[95] 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.

[96] Giorgio Giacinto, Fabio Roli, and Giorgio Fumera. Design of effective multiple classifier systems by clustering of classifiers. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 2, pages 160–163. IEEE, 2000.

[97] Jing Gu, Licheng Jiao, Fang Liu, Shuyuan Yang, Rongfang Wang, Puhua Chen, Yuanhao Cui, Junhu Xie, and Yake Zhang. Random subspace based ensemble sparse representation. *Pattern Recognition*, 74:544–555, 2018.

[98] Bogdan Gulowaty and Paweł Ksieniewicz. Smote algorithm variations in balancing data streams. In Hujun Yin, David Camacho, Peter Tino, Antonio J. Tallón-Ballesteros, Ronaldo Menezes, and Richard Allmendinger, editors, *Intelligent Data Engineering and Automated Learning – IDEAL 2019*, pages 305–312, Cham, 2019. Springer International Publishing.

[99] Isabelle Guyon. Design of experiments of the nips 2003 variable selection benchmark. In *NIPS 2003 workshop on feature extraction and feature selection*, 2003.

[100] H. Han, W. Wang, and B. Mao. Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In *Advances in Intelligent Computing, International Conference on Intelligent Computing 2005, Hefei, China, August 23-26, 2005, Proceedings, Part I*, pages 878–887, 2005.

[101] David J Hand. Measuring classifier performance: a coherent alternative to the area under the roc curve. *Machine learning*, 77(1):103–123, 2009.

[102] Peter Hart. The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory*, 14(3):515–516, 1968.

[103] Trevor Hastie, Robert Tibshirani, et al. Classification by pairwise coupling. *Annals of statistics*, 26(2):451–471, 1998.

[104] H. He, Y. Bai, E. A. Garcia, and S. Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Proceedings of the International Joint Conference on Neural Networks, 2008, part of the IEEE World Congress on Computational Intelligence, 2008, Hong Kong, China, June 1-6, 2008*, pages 1322–1328, 2008.

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

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

[107] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

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

[109] Tin Kam Ho, J. J. Hull, and S. N. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–75, Jan 1994.

[110] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*, pages 409–426. Springer, 1994.

[111] Myles Hollander, Douglas A Wolfe, and Eric Chicken. *Nonparametric statistical methods*, volume 751. John Wiley & Sons, 2013.

[112] Yu Hen Hu and Jeng-Neng Hwang. Handbook of neural network signal processing, 2002.

[113] Konrad Jackowski, Bartosz Krawczyk, and Michal Woźniak. Improved adaptive splitting and selection: the hybrid training method of a classifier based on a feature space partitioning. *Int. J. Neural Syst.*, 24(3), 2014.

[114] Robert A Jacobs. Methods for combining experts' probability assessments. *Neural computation*, 7(5):867–888, 1995.

[115] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[116] Anil K Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1):4–37, 2000.

[117] Hamid Jamalinia, Saber Khalouei, Vahideh Rezaie, Samad Nejatian, Karamolah Bagheri-Fard, and Hamid Parvin. Diverse classifier ensemble creation based on heuristic dataset modification. *Journal of Applied Statistics*, 45(7):1209–1226, 2018.

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

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

[120] John D. Kelleher, Brian Mac Namee, and Aoife D'Arcy. *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. The MIT Press, 2015.

[121] Ross D King, Kenneth E Whelan, Ffion M Jones, Philip GK Reiser, Christopher H Bryant, Stephen H Muggleton, Douglas B Kell, and Stephen G Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247–252, 2004.

[122] Josef Kittler, Mohamad Hatef, Robert PW Duin, and Jiri Matas. On combining classifiers. *IEEE transactions on pattern analysis and machine intelligence*, 20(3):226–239, 1998.

[123] Albert H.R. Ko, Robert Sabourin, and Jr. Alceu Souza Britto. From dynamic classifier selection to dynamic ensemble selection. *Pattern Recognition*, 41(5):1718 – 1731, 2008.

[124] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[125] Ron Kohavi and David Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, pages 275–283, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[126] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8:2755–2790, 2007.

[127] J Zico Kolter and Marcus A Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *The Journal of Machine Learning Research*, 8:2755–2790, 2007.

[128] Joanna Komorniczak, Paweł Zyblewski, and Paweł Ksieniewicz. Prior probability estimation in dynamically imbalanced data streams. In *The International Joint Conference on Neural Networks*, 2021.

[129] Łukasz Korycki and Bartosz Krawczyk. Combining active learning and self-labeling for data stream mining. In Marek Kurzynski, Michal Wozniak, and Robert Burduk, editors, *Proceedings of the 10th International Conference on Computer Recognition Systems CORES 2017*, pages 481–490, Cham, 2018. Springer International Publishing.

[130] Michał Koziarski, Bartosz Krawczyk, and Michał Woźniak. Radial-based approach to imbalanced data oversampling. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 318–327. Springer, 2017.

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

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

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

[134] Bartosz Krawczyk and Alberto Cano. Adaptive ensemble active learning for drifting data stream mining. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 2763–2771. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[135] Bartosz Krawczyk, Leandro L. Minku, Joao Gama, Jerzy Stefanowski, and Michał Woźniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37(Supplement C):132 – 156, 2017.

[136] Bartosz Krawczyk, Bernhard Pfahringer, and Michal Wozniak. Combining active learning with concept drift detection for data stream mining. In Naoki Abe, Huan Liu, Calton Pu, Xiaohua Hu, Nesreen Ahmed, Mu Qiao, Yang Song, Donald Kossmann, Bing Liu, Kisung Lee, Jiliang Tang, Jingrui He, and Jeffrey S. Saltz, editors, *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*, pages 2239–2244. IEEE, 2018.

[137] Bartosz Krawczyk, Michal Wozniak, and Boguslaw Cyganek. Clustering-based ensembles for one-class classification. *Information Sciences*, 264:182–195, 2014.

[138] 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.

[139] P. Ksieniewicz, P. Zyblewski, M. Choraś, R. Kozik, A. Giełczyk, and M. Woźniak. Fake news detection from data streams. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020.

[140] Pawel Ksieniewicz. Undersampled majority class ensemble for highly imbalanced binary classification. In *Proceedings of the Second International Workshop on Learning with Imbalanced Domains: Theory and Applications*, volume 94 of *Proceedings of Machine Learning Research*, pages 82–94, ECML-PKDD, Dublin, Ireland, 10 Sep 2018. PMLR.

[141] Paweł Ksieniewicz and Paweł Zyblewski. stream-learn–open-source python library for difficult data stream batch analysis. *arXiv preprint arXiv:2001.11077*, 2020.

[142] Paweł Ksieniewicz and Paweł Zyblewski. stream-learn–open-source python library for difficult data stream batch analysis. *Neurocomputing*, 2021.

[143] Paweł Ksieniewicz, Paweł Zyblewski, Weronika Borek, Rafał Kozik, Michał Choras, and Michał Wozniak. Alphabet flatting as a variant of n-gram feature extraction method in ensemble classification of fake news. *Engineering Applications of Artificial Intelligence*, 2021.

[144] Paweł Ksieniewicz, Paweł Zyblewski, and Robert Burduk. Fusion of linear base classifiers in geometric space. *Knowledge-Based Systems*, 2021.

[145] Paweł Ksieniewicz, Michał Woźniak, Bogusław Cyganek, Andrzej Kasprzak, and Krzysztof Walkowiak. Data stream classification using active learned neural networks. *Neurocomputing*, 353:74 – 82, 2019. Recent Advancements in Hybrid Artificial Intelligence Systems.

[146] Miroslav Kubat, Robert C. Holte, and Stan Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30(2):195–215, Feb 1998.

[147] Miroslav Kubat and Stan Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In *In Proceedings of the 14th International Conference on Machine Learning*, pages 179–186. Morgan Kaufmann, 1997.

[148] Ludmila I. Kuncheva. Clustering-and-selection model for classifier combination. In *Fourth International Conference on Knowledge-Based Intelligent Information Engineering Systems & Allied Technologies, KES 2000, Brighton, UK, 30 August - 1 September 2000, Proceedings, 2 Volumes*, pages 185–188, 2000.

[149] Ludmila I. Kuncheva. Classifier ensembles for changing environments. In *Multiple Classifier Systems, 5th International Workshop, MCS 2004, Cagliari, Italy, June*

9-11, 2004, Proceedings, volume 3077 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2004.

[150] Ludmila I Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms.* John Wiley & Sons, Hoboken, NJ, 2004.

[151] 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.

[152] B. Kurlej and M. Woźniak. Active learning approach to concept drift problem. *Logic Journal of the IGPL*, 20(3):550–559, 2012.

[153] Krishna K Ladha. The condorcet jury theorem, free speech, and correlated votes. *American Journal of Political Science*, pages 617–634, 1992.

[154] Mateusz Lango and Jerzy Stefanowski. Multi-class and feature selection extensions of roughly balanced bagging for imbalanced data. *Journal of Intelligent Information Systems*, 50(1):97–127, 2018.

[155] Jorma Laurikkala. Improving identification of difficult small classes by balancing class distribution. In *Conference on Artificial Intelligence in Medicine in Europe*, pages 63–66. Springer, 2001.

[156] A. Lazarevic and Z. Obradovic. The effective pruning of neural network classifiers. 2001 IEEE/INNS International Conference on Neural Networks, IJCNN 2001, 2001.

[157] Aleksandar Lazarevic and Vipin Kumar. Feature bagging for outlier detection. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 157–166, 2005.

[158] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer, 1994.

[159] Ming Li and Zhi-Hua Zhou. Setred: Self-training with editing. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 611–621. Springer, 2005.

[160] Ryan N. Lichtenwalter and Nitesh V. Chawla. *Adaptive Methods for Classification in Arbitrarily Imbalanced and Drifting Data Streams*, pages 53–75. Springer, Berlin, Heidelberg, 2010.

[161] Nicholas Littlestone and Marcus K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

[162] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[163] V. Lopez, A. Fernandez, J. G. Moreno-Torres, and F. Herrera. Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. open problems on intrinsic data characteristics. *Expert Systems with Applications*, 39(7):6585–6608, 2012.

[164] 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.

[165] Rafal Lysiak, Marek Kurzynski, and Tomasz Woloszynski. Optimal selection of ensemble classifiers using measures of competence and diversity of base classifiers. *Neurocomputing*, 126:29–35, 2014.

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

[167] T. Maciejewski and J. Stefanowski. Local neighbourhood extension of SMOTE for mining imbalanced data. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining 2011, part of the IEEE Symposium Series on Computational Intelligence 2011, April 11-15, 2011, Paris, France*, pages 104–111, 2011.

[168] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[169] Dragos D. Margineantu and Thomas G. Dietterich. Pruning adaptive boosting. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 211–218, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[170] Oden Maron and Andrew W Moore. The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, 11(1):193–225, 1997.

[171] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[172] Fernanda Li Minku, Hirotaka Inoue, and Xin Yao. Negative correlation in incremental learning. *Natural Computing*, 8(2):289–320, 2009.

[173] Marvin Minsky. *Society of mind*. Simon and Schuster, 1988.

[174] Tom M Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math Boston, 1997.

[175] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[176] Jose García Moreno-Torres, José A Sáez, and Francisco Herrera. Study on the impact of partition-induced dataset shift on k-fold cross-validation. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1304–1312, 2012.

[177] Yi L Murphey, Haoxing Wang, Guobin Ou, and Lee A Feldkamp. Oaho: an effective algorithm for multi-class learning from imbalanced data. In *2007 International Joint Conference on Neural Networks*, pages 406–411. IEEE, 2007.

[178] 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.

[179] Krystyna Napierala and Jerzy Stefanowski. Types of minority class examples and their influence on learning classifiers from imbalanced data. *Journal of Intelligent Information Systems*, 46:563–597, 2015.

[180] Krystyna Napierała, Jerzy Stefanowski, and Szymon Wilk. Learning from imbalanced data in presence of noisy and borderline examples. In *International Conference on Rough Sets and Current Trends in Computing*, pages 158–167. Springer, 2010.

[181] H. M. Nguyen, E. W. Cooper, and K. Kamei. Online learning from imbalanced data streams. In *International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, pages 347–352, Oct 2011.

[182] Hien M Nguyen, Eric W Cooper, and Katsuari Kamei. Borderline over-sampling for imbalanced data classification. *International Journal of Knowledge Engineering and Soft Data Paradigms*, 3(1):4–21, 2011.

[183] Dayvid V. R. Oliveira, George D. C. Cavalcanti, and Robert Sabourin. Online pruning of base classifiers for dynamic ensemble selection. *Pattern Recognition*, 72:44–58, 2017.

[184] N. C. Oza. Online bagging and boosting. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2340–2345 Vol. 3, Oct 2005.

[185] Nikunj C. Oza and Stuart Russell. Online bagging and boosting. In *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics (AISTATS'01)*, page 105112, Key West, USA, 2001. Morgan Kaufmann.

[186] Paul Pavlidis, Jason Weston, Jinsong Cai, and William Noble Grundy. Gene functional classification from heterogeneous data. In *Proceedings of the fifth annual international conference on Computational biology*, pages 249–255, 2001.

[187] 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.

[188] Felipe Pinagé, Eulanda M dos Santos, and João Gama. A drift detection method based on dynamic classifier selection. *Data Mining and Knowledge Discovery*, 34(1):50–74, 2020.

[189] Robi Polikar, Lalita Upda, Satish S Upda, and Vasant 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.

[190] David Powers and Ailab. Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. *J. Mach. Learn. Technol*, 2:2229–3981, 01 2011.

[191] Mahardhika Pratama, Jie Lu, Edwin Lughofer, Guangquan Zhang, and Sreenatha Anavatti. Scaffolding type-2 classifier for incremental learning under concept drifts. *Neurocomputing*, 191:304–329, 2016.

[192] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.

[193] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[194] Lior Rokach. *Pattern classification using ensemble methods*, volume 75. World Scientific, 2010.

[195] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[196] A. Roy, R. M. O. Cruz, R. Sabourin, and G. D. C. Cavalcanti. Meta-regression based pool size prediction scheme for dynamic selection of classifiers. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 216–221, 2016.

[197] Anandarup Roy, Rafael M.O. Cruz, Robert Sabourin, and George D.C. Cavalcanti. Meta-learning recommendation of default size of classifier pool for meta-des. *Neurocomputing*, 216:351–362, 2016.

[198] Anandarup Roy, Rafael M.O. Cruz, Robert Sabourin, and George D.C. Cavalcanti. A study on combining dynamic selection and data preprocessing for imbalance learning. *Neurocomputing*, 286:179 – 192, 2018.

[199] Rui Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[200] Stuart J Russell and Peter Norvig. Artificial intelligence-a modern approach, third international edition., 2010.

[201] D. Ruta and B. Gabrys. A theoretical analysis of the limits of majority voting errors for multiple classifier systems. *Pattern Analysis and Applications*, 2(4):333–350, 2002.

[202] Dymitr Ruta and Bogdan Gabrys. Classifier selection for majority voting. *Information Fusion*, 6(1):63–81, 2005.

[203] M. Sabourin, A. Mitiche, D. Thomas, and G. Nagy. Classifier combination for hand-printed digit recognition. In *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR '93)*, pages 163–166, 1993.

[204] Şenay Yaşar Sağlam and W Nick Street. Distant diversity in dynamic class prediction. *Annals of Operations Research*, 263(1):5–19, 2018.

[205] Yutaka Sasaki. The truth of the f-measure. *Teach Tutor Mater*, 01 2007.

[206] Giovanni Seni and John F Elder. Ensemble methods in data mining: improving accuracy through combining predictions. *Synthesis lectures on data mining and knowledge discovery*, 2(1):1–126, 2010.

[207] Burr Settles. *Active Learning*. Morgan & Claypool Publishers, 2012.

[208] Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pages 1289–1296, 2007.

[209] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992.

[210] 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.

[211] J. Shan, H. Zhang, W. Liu, and Q. Liu. Online active learning ensemble framework for drifted data streams. *IEEE Transactions on Neural Networks and Learning Systems*, 30(2):486–498, Feb 2019.

[212] Hanan Shteingart, Eran Marom, Igor Itkin, Gil Shabat, Michael Kolomenkin, Moshe Salhov, et al. Majority voting and the condorcet's jury theorem. *arXiv preprint arXiv:2002.03153*, 2020.

[213] 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. Citeseer, 1996.

[214] R. G. F. Soares, A. Santana, A. M. P. Canuto, and M. C. P. de Souto. Using accuracy and diversity to select classifiers to build ensembles. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 1310–1316, July 2006.

[215] Tsu T Soong. *Fundamentals of probability and statistics for engineers.* John Wiley & Sons, 2004.

[216] Carlos Oscar Sánchez Sorzano, Javier Vargas, and A Pascual Montano. A survey of dimensionality reduction techniques. *arXiv preprint arXiv:1403.2877*, 2014.

[217] Mariana A. Souza, George D.C. Cavalcanti, Rafael M.O. Cruz, and Robert Sabourin. Online local pool generation for dynamic classifier selection. *Pattern Recognition*, 85:132–148, 2019.

[218] V. M. A. Souza, D. M. Reis, A. G. Maletzke, and G. E. A. P. A. Batista. Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, 34:1805–1858, 2020.

[219] Katarzyna Stapor, Paweł Ksieniewicz, Salvador Garcia, and Michał Woźniak. How to design the fair experimental classifier evaluation. *Applied Soft Computing*, page 107219, 2021.

[220] Jerzy Stefanowski and Szymon Wilk. Selective pre-processing of imbalanced data for improving classification performance. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 283–292. Springer, 2008.

[221] Nick Street and Y. Kim. A streaming ensemble algorithm (sea) for large-scale classification. *Proceedings of the 7Th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 377–382, 01 2001.

[222] Dominika Sułot, Paweł Zyblewski, and Paweł Ksieniewicz. Analysis of variance application in the construction of classifier ensemble based on optimal feature subset for the task of supporting glaucoma diagnosis. In *International Conference on Computational Science*. Springer, 2021.

[223] Y. Sun, K. Tang, L. L. Minku, S. Wang, and X. Yao. Online ensemble learning of data streams with gradually evolved classes. *IEEE Transactions on Knowledge and Data Engineering*, 28(6):1532–1545, June 2016.

[224] Y. Sun, A. K. C. Wong, and M. S. Kamel. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(4):687–719, 2009.

[225] Yanmin Sun, Mohamed S. Kamel, Andrew K.C. Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358 – 3378, 2007.

[226] James Surowiecki. *The wisdom of crowds*. Anchor, 2005.

[227] John A Swets. Measuring the accuracy of diagnostic systems. *Science*, 240(4857):1285–1293, 1988.

[228] David MJ Tax and Robert PW Duin. Using two-class classifiers for multiclass classification. In *Object recognition supported by user interaction for service robots*, volume 2, pages 124–127. IEEE, 2002.

[229] Ivan Tomek. Two modifications of CNN. *IEEE Transactions on Systems, Man, and Cybernetics*, 6:769–772, 1976.

[230] Isaac Triguero, Salvador García, and Francisco Herrera. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information systems*, 42(2):245–284, 2015.

[231] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. In *European Conference on Machine Learning*, pages 465–476. Springer, 2004.

[232] Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10(66-71):13, 2009.

[233] V Vapnik. Statistical learning theory new york. *NY: Wiley*, 1998.

[234] Vladimir Vapnik. Estimation of dependences based on empirical data: Springer series in statistics (springer series in statistics), 1982.

[235] Vladimir N Vapnik. The nature of statistical learning theory, 1995.

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

[237] F. N. Walmsley, G. D. C. Cavalcanti, D. V. R. Oliveira, R. M. O. Cruz, and R. Sabourin. An ensemble generation method based on instance hardness. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018.

[238] Dayong Wang, Pengcheng Wu, Peilin Zhao, Yue Wu, Chunyan Miao, and Steven CH Hoi. High-dimensional data stream classification via sparse online learning. In *2014 IEEE international conference on data mining*, pages 1007–1012. IEEE, 2014.

[239] 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, pages 226–235, New York, NY, USA, 2003. ACM.

[240] Qili Wang, Wei Xu, and Han Zheng. Combining the wisdom of crowds and technical analysis for financial market prediction using deep random subspace ensembles. *Neurocomputing*, 299:51–61, 2018.

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

[242] S. Wang, L. L. Minku, and X. Yao. A learning framework for online class imbalance learning. In *IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL)*, pages 36–45, April 2013.

[243] S. Wang, L. L. Minku, and X. Yao. Resampling-based ensemble methods for online class imbalance learning. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1356–1368, May 2015.

[244] Senzhang Wang, Zhoujun Li, Wenhan Chao, and Qinghua Cao. Applying adaptive over-sampling technique based on data density and cost-sensitive svm to imbalanced learning. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2012.

[245] Shuo Wang, Leandro L. Minku, and Xin Yao. A systematic study of online class imbalance learning with concept drift. *CoRR*, abs/1703.06683, 2017.

[246] Yi Wang, Yang Zhang, and Yong Wang. *Mining Data Streams with Skewed Distribution by Static Classifier Ensemble*, pages 65–71. Springer, Berlin, Heidelberg, 2009.

[247] David Williams, Xuejun Liao, Ya Xue, Lawrence Carin, and Balaji Krishnapuram. On classification with incomplete data. *IEEE transactions on pattern analysis and machine intelligence*, 29(3):427–436, 2007.

[248] Dennis L Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, 2(3):408–421, 1972.

[249] Tomasz Woloszynski and Marek Kurzynski. A probabilistic model of classifier competence for dynamic ensemble selection. *Pattern Recognition*, 44(10-11):2656–2668, 2011.

[250] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[251] David H Wolpert. The supervised learning no-free-lunch theorems. In *Soft computing and industry*, pages 25–42. Springer, 2002.

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

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

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

[255] Michal Wozniak and Konrad Jackowski. Some remarks on chosen methods of classifier fusion based on weighted voting. In Emilio Corchado, Xindong Wu, Erkki Oja, Álvaro Herrero, and Bruno Baruque, editors, *Hybrid Artificial Intelligence Systems*, pages 541–548, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[256] Michał Woźniak, Andrzej Kasprzak, and Piotr Cal. Weighted aging classifier ensemble for the incremental drifted data streams. In *Flexible Query Answering Systems*, pages 579–588, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[257] Michal Wozniak and Marcin Zmyslony. Combining classifiers using trained fuser—analytical and experimental results. *Neural Network World*, 20(7):925, 2010.

[258] Yufei Xia, Chuanzhe Liu, Bowen Da, and Fangming Xie. A novel heterogeneous ensemble credit scoring model based on bstacking approach. *Expert Systems with Applications*, 93:182–199, 2018.

[259] Jin Xiao, Ling Xie, Changzheng He, and Xiaoyi Jiang. Dynamic classifier ensemble model for customer classification with imbalanced class distribution. *Expert Systems with Applications*, 39(3):3668 – 3675, 2012.

[260] X. Xiaolong, C. Wen, and S. Yanfei. Over-sampling algorithm for imbalanced data classification. *Journal of Systems Engineering and Electronics*, 30(6):1182–1191, 2019.

[261] Kaixiang Yang, Zhiwen Yu, Xin Wen, Wenming Cao, CL Philip Chen, Hau-San Wong, and Jane You. Hybrid classifier ensemble for imbalanced data. *IEEE transactions on neural networks and learning systems*, 31(4):1387–1400, 2019.

[262] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196, 1995.

[263] Show-Jane Yen and Yue-Shi Lee. Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset. In *Intelligent Control and Automation*, pages 731–740. Springer, 2006.

[264] Shin-ichi Yoshida, Kohei Hatano, Eiji Takimoto, and Masayuki Takeda. Adaptive online prediction using weighted windows. *IEICE TRANSACTIONS on Information and Systems*, 94(10):1917–1923, 2011.

[265] Hualong Yu, Xibei Yang, Shang Zheng, and Changyin Sun. Active learning from imbalanced data: A solution of online weighted extreme learning machine. *IEEE Transactions on Neural Networks and Learning Systems*, 30(4):1088–1103, 2018.

[266] George Udny Yule. On the association of attributes in statistics: with illustrations from the material of the childhood society, &c. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 194(252-261):257–319, 1900.

[267] H. Zhang and L. Cao. A spectral clustering based ensemble pruning approach. *Neurocomputing*, 139:289–297, 2014.

[268] Hang Zhang, Weike Liu, Jicheng Shan, and Qingbao Liu. Online active learning paired ensemble for concept drift and class imbalance. *IEEE Access*, 6:73815–73828, 2018.

[269] 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.

[270] Z. H. Zhou. *Ensemble Methods: Foundations and Algorithms.* Chapman & Hall CRC, Boca Raton, FL, 2012.

[271] 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.

[272] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: Many could be better than all. *Artif. Intell.*, 137(1-2):239–263, May 2002.

[273] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

[274] I. Zliobaite, A. Bifet, B. Pfahringer, and G. Holmes. Active learning with drifting streaming data. *IEEE Trans. Neural Netw. Learning Syst.*, 25(1):27–39, 2014.

[275] Weiwei Zong, Guang-Bin Huang, and Yiqiang Chen. Weighted extreme learning machine for imbalance learning. *Neurocomput.*, 101:229–242, February 2013.

[276] Paweł Zyblewski. Clustering-based ensemble pruning in the imbalanced data classification. In *International Conference on Computational Science*. Springer, 2021.

[277] Paweł Zyblewski, Paweł Ksieniewicz, and Michał Woźniak. Classifier selection for highly imbalanced data streams with minority driven ensemble. In *Artificial Intelligence and Soft Computing*, pages 626–635, Cham, 2019. Springer International Publishing.

[278] Paweł Zyblewski, Paweł Ksieniewicz, and Michał Woźniak. Combination of active and random labeling strategy in the non-stationary data stream classification. In *International Conference on Artificial Intelligence and Soft Computing*, pages 576–585. Springer, 2020.

[279] Paweł Zyblewski, Marek Pawlicki, Rafał Kozik, and Michał Choras. Cyber-attack detection from iot benchmark considered as data streams. In *International Conference on Computer Recognition Systems*, 2021.

[280] Paweł Zyblewski, Robert Sabourin, and Michał Woźniak. Data preprocessing and dynamic ensemble selection for imbalanced data stream classification. In *Machine Learning and Knowledge Discovery in Databases*, pages 367–379, Cham, 2020. Springer International Publishing.

[281] Paweł Zyblewski and Michał Woźniak. Clustering-based ensemble pruning and multistage organization using diversity. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 287–298. Springer, 2019.

[282] Paweł Zyblewski and Michał Woźniak. Dynamic classifier selection for data with skewed class distribution using imbalance ratio and euclidean distance. In *International Conference on Computational Science*, pages 59–73. Springer, 2020.

[283] Paweł Zyblewski and Michał Woźniak. Novel clustering-based pruning algorithms. *Pattern Analysis and Applications*, pages 1–10, 2020.

[284] Paweł Zyblewski, Robert Sabourin, and Michał Woźniak. Preprocessed dynamic classifier ensemble selection for highly imbalanced drifted data streams. *Information Fusion*, 66:138 – 154, 2021.