Subspace-based dynamic selection for high-dimensional data

Submitted October 2022, in partial fulfillment of
the conditions for the award of the degree **Doctor of Philosophy in Computer Science.**

Alexandre Maciel Guerra
ID 14290661

Supervised by Jamie Twycross and Grazziela Figueredo

School of Computer Science
University of Nottingham

I hereby declare that this dissertation is all my own work

Signature: Alexandre Maciel Guerra

Date: 15/10/2022

Abstract

The number of features collected has increased greatly in the past decade, particularly in medicine and life sciences, which brings challenges and opportunities. Making reliable predictions, exploring associations and extracting meaningful information in high-dimensional data are some of the problems that are yet to be solved. Due to intrinsic properties of high-dimensional spaces such as distance concentration and hubness, traditional classification and clustering algorithms face difficult challenges. In general, a Multiple Classifier System (MCS) provides better classification accuracy than individual classifiers. One of the most promising approaches to MCS is Dynamic Selection (DS) methods, which work by selecting classifiers on the fly, according to each unknown test sample. The rationale behind this is that not every classifier is an expert in predicting all samples, rather each classifier or a combination of classifiers is an expert in a different region of the feature space; whose quality can significantly impact the overall performance.

This thesis provides three major contributions. First, traditional DS methods fail to perform effectively in high-dimensional data sets due to the use of a k-Nearest Neighbour (k-NN) to define the region competence and, moreover, they do not indicate which are the most important features for classification. Second, two frameworks were proposed the Subspace-Based Dynamic Selection (SBDS) and the Classifier SBDS (cSBDS) which integrate characteristics of DS methods and subspace clustering. Subspace clustering methods localise their search for clusters and are able to uncover clusters that exist in multiple, possible overlapping subspaces of features and/or samples. The subspace clustering approach separates the high-dimensional feature space into small feature spaces with a reduced number of features and samples in each one. The results indicate that the cSBDS framework performs statistically better when compared to DS methods and majority voting on real-world and synthetic datasets. Third, we provide a comparison between the features selected by the cSBDS framework and feature importance methods. The results indicate that for high-dimensional datasets, the cSBDS framework is able to capture the most important features when the number of clusters per class is increased, while traditional feature importance methods lose this capability.

Acknowledgements

“Those who pass by us, do not go alone, and do not leave us alone; they leave a bit of themselves, and take a little of us”
(Antoine de Saint-Exupéry)

The past years have been the most arduous journey of challenges, constructions and maturation of my career. In this period, I learned that a thesis is the extension of the author’s life. That to build something is not enough just all my willpower, but all the support of the people who are by my side. For this reason, I sincerely and deeply thank all the people who have passed through my life and who have encouraged and helped me to become a better person and a better researcher.

“And I learned that one always depends
On so many, many, different people
Every person is always the marks
of the daily lessons of so many other people.
It’s so beautiful when we understand
That we are so many people
Wherever we go.
It’s so beautiful when you feel
That you’re never alone
No matter how much you think you are...”
(Caminhos do Coração – Gonzaguinha.)

First, I would like to thank my supervisors Dr Jamie Twycross and Dr Grazziela P. Figueredo for all your dedication, patience and for believing in me. For having accepted me for this project even knowing all the difficulties of me being in a different country. For for constantly providing support throughout my PhD, especially though difficult periods some of which you were familiar with, and in other moments like the COVID-19 pandemic where no one was prepared for. You have always believed in me. You have always helped me in my brightest and in my toughest moments. You are certainly a model of researchers and teachers whom I have the pleasure of following and in whom I will try to mirror myself.

Thanks also to my fiancee Iracema, who has accompanied me throughout the course of this PhD, always helping, calming and encouraging me. For two and half years we were separated by an ocean, and during this time you were able to comfort me and show me that love is not measured by distance but the feelings we have for one another. Then during the pandemic years you have shown me your strength helping patients in need, demonstrating how life can be fragile but how important support and careness are in the most difficult times. Thank you for staying by my side in the good and not so good times. For always believing in me, for making me a better person and for being a model of dedication and goodwill.

I would also like to thank my parents Gil and Andrea, my sister Helena and my family for always being by my side, helping me, encouraging me and keeping my feet on the ground. Specially, I would like to thank my grandmother Olga and my uncle Sergio who helped raising me during my childhood and who are no longer physically present among us, they are certainly right here, by my side, experiencing this moment with great pride; and may God bless you always.

Thanks to all the members of the School of Computer Science with whom I worked during this period. I also express my enormous gratitude to Dr Tania Dottorini and her team for working with me and supporting me throughout all these years.

As I undertake my corrections following my viva, I want to take a moment to thank my examiners Prof. Alceu de Souza Britto Junior and Dr Isaac Triguero Velazquez for their attention, comments and advises that undoubtedly improved my thesis in all possible aspects.

One more step has been completed! I know it’s just the beginning... and that soon another long stage will begin. And all the lessons I learned will help me and give me strength and confidence to follow this path.

List of Tables

***

List of Figures

***

List of Algorithms

***

Acronyms

***

BKS	Behaviour Knowledge Space.
CR	Classifier Rank.
cSBDS	Classifier SBDS.
DCS	Dynamic Classifier Selection.
DCS-LA	DCS by Local Accuracy.
DES	Dynamic Ensemble Selection.
DES-EXP	DES - Exponential.
DES-KL	DES - Kullback-Leibler Divergence.
DES-kMeans	DES - k-Means.
DES-kNN	DES - K-Nearest Neighbour.
DES-MD	DES - Minimal Difference.
DES-P	DES - Performance.
DES-RRC	DES - Randomised Reference Classifier.
DS	Dynamic Selection.
DSOC	Dynamic Selection on Complexity.
GKDE	Gaussian Kernel Density Estimator.

IBH	Insect Byte Hypersensitivity.
k-NN	*k*-Nearest Neighbour.
KNOP	*k*-Nearest Output Profiles.
KNOP-E	KNOP - Eliminate.
KNOP-U	KNOP - Union.
KNORA	*k*-Nearest ORAcles.
KNORA-E	KNORA - Eliminate.
KNORA-U	KNORA - Union.
LCA	Local Class Accuracy.
MCB	Multiple Classifier Behaviour.
MCR	Modified Classifier Rank.
MCS	Multiple Classifier System.
Meta-DES	Meta-learning - DES.
MLA	Modified Local Accuracy.
MLP	Multiple Layer Perceptron Neural Network.
NB	Naive Bayes classifier.
NBP	Jenks Natural Break Points.
OLA	Overall Local Accuracy.
RBF-SVM	Radial Basis Function Support Vector Machine.
RF	Random Forest classifier.
RofC	Region of Competence.

RSS Random Subspace Sampling.

SBDS Subspace-Based Dynamic Selection.

SS Static Selection.

SVM Support Vector Machine.

Contents

***

Chapter 1

Introduction

***

1.1 Introduction

Classification is an important task in pattern recognition, which is one of the main reasons why the number of methods developed have dramatically increased in the past decades [1, 2]. Even though the methods developed may differ in different aspects and achieve the objectives from which they were proposed, creating a single classifier to cover all the variability in most pattern recognition problem is still a challenge, even more with the increase in the number of features in some areas like medicine and biology [1, 2].

For these reasons, Multiple Classifier System (MCS) are a very active area of research in classification problems. Recently, several studies published results demonstrating its advantages over a single robust classifier [3–5]. The idea behind MCS relies on the fact that a combination of "different" classifiers might have a strong degree of "independence" in the errors, *i.e.* make few coincident errors. Thus, the errors committed by a classifier can be overridden by the correct classification of other classifiers. Static Selection (SS), Dynamic Classifier Selection (DCS) and Dynamic Ensemble Selection (DES) are the techniques commonly employed to determine the set of classifiers within the ensemble [4, 6]. SS works by selecting a group of classifiers for all new samples, while DCS and DES select a single or a group of classifiers for each new sample, respectively. Recently, Dynamic Selection (DS) methods have been preferred over static methods due to their ability to create different classifier configurations, i.e. different groups of classifiers are experts in different local regions of the feature space. As for many cases, different samples are associated with different classification difficulties and the ability to choose a group of classifiers can possibly overcome static selection methods limitations [4, 6, 7].

However, traditional DS methods fail to perform effectively in high-dimensional datasets with more features than samples [8, 9]. This is caused by the use of nearest-neighbours approaches to define the region competence. Another limitation of those methods is that they do not provide any indication of how important the data features are for a classification task. The focus of our research is to identify ways to overcome these limitations. Therefore, this thesis proposes alternative ways to replace k-Nearest Neighbour (k-NN) in the DS framework. In practical problems, different query samples have different classification difficulties and, in high-dimensional datasets, may be located in different subspace clusters [4, 10]. Hence, it is intuitive to think that adopting different subspaces to predict the pattern of different test samples may increase the performance of a multiple classifier system. Moreover, by using the concepts of subspace clustering into the DS framework, it would be possible to know which features are more important for the classification of each test pattern. This chapter aims to outline the motivations, research gaps and aim of the thesis and to introduce the thesis structure.

1.2 Background and Motivation

The quantity of data collected from multiple sources have increased significantly in the past decade . Heterogeneity, scalability, computational time and complexity are challenges that impede progress to extract meaningful information from data [11, 12]. Datasets with more features than samples are typical in some domains, such as biology, medicine, bioinformatics and neuroimaging. Often in these areas data instances do not exist in abundance or are expensive to acquire, although technology allows for the acquisition of multiple features for each observation [13]. Patient data is an example where the number of samples are limited to the number of patients in a practice, but there is a significant higher amount of information (visits, diagnoses, interventions, laboratory tests, clinical narratives, image records, patient history, demographic attributes. etc.) [14, 15]. DNA microarray is another example of these types of datasets. Data collected from tissue and cell samples are used to measure the levels of gene expression. The number of genes is usually far higher than the number of patients [16].

Exploring associations, making reliable predictions and extracting information are problems yet to be solved in high-dimensional data [11, 15]. Traditional machine learning techniques were often created having in mind intuitive properties and examples in low-dimensional datasets and when these methods are applied to high-dimensional datasets they might not behave as expected [17–19]. Collinearity, numerical instability, overfitting, model instability are some of the known problems that can occur when analysing high-dimensional datasets. Moreover, high-dimensional spaces have geometrical properties that are not intuitive [17], for instance:

- *Distance concentration*: which shows the tendency of the distance between all points to become almost equal, making, therefore, nearest neighbours to be meaningless [20].
- *Hubness*: the tendency of high-dimensional data to contain points (hubs) that frequently occur in k-nearest neighbour lists of other points [20].

Concentration of distances, a phenomenon related to hubness, was studied on high-dimensional datasets for general distance measures [21, 22], for Minkowski and fractional distances [23–26] and for the cosine distance [27]. Therefore, the issues surrounding high-dimensional datasets are still present regardless of the distance metric used as indicated by Radovanovic *et al.* [20], even though one distance metric may perform better in a specific dataset.

For any classifier to be successful, it is usually necessary to have sufficient data to cover the feature space during training [16, 17]. In general, ensemble classifiers provide better classification accuracy than individual classifiers [28]. However, many ensemble methods proposed in the literature do not perform well in terms of accuracy in high-dimensional biomedicine data, according to a survey made by Meshram and Shinde (2015) [28].

Multiple classifier systems (MCSs) are widely researched in machine learning and pattern recognition. Several studies published results demonstrating its advantages over a single robust classifier [3–5]. DS is one of the most promising approaches to MCS [4, 5], with many researchers reporting their superior performance over a single robust classifier and other MCS approaches (majority voting, bagging, boosting) [4–7, 29, 30]. DS techniques can select either a single classifier (*Dynamic Classifier Selection*) or an ensemble of classifiers (*Dynamic Ensemble Selection*). These techniques are used to select classifiers based on their competence level to predict the label of a test sample. The competence is estimated considering only the samples of a local region of the feature space where the test sample is located (region of competence). The majority of DS techniques rely on k-Nearest Neighbour (k-NN) algorithms and the quality of the neighbourhood impacts on their performance [4–6].

Intuition tells that a good pool of classifiers is formed by methods with high accuracy and as much diverse as possible. However, diversity is not an exact concept and defining it is not trivial [31, 32]. A point of consensus is that classifiers that make statistically different errors when combined have the power to increase the overall performance. Bagging, boosting and Random Subspace Sampling (RSS) are some of the methods that can be used to create a diverse pool of classifiers [7]. The first two create pools based on classifiers being trained on different sets of samples. While the last one, creates diversity by training classifiers in different sets of features that are randomly selected without repetition [7]. Because RSS chooses the subspaces randomly, it is unfeasible to understand how one features relates to another and what are the reasons behind the selection of features.

In general, ensemble classifiers provide a better classification performance than individual classifiers [28]. Moreover, individual classifiers are usually not able to handle the noise and imbalance data in high-dimensional datasets [28, 33]. In 2015, Meshram & Shinde [28] presented a survey on ensemble methods for high-dimensional datasets highlighting their limitations, some of the limitations for three state-of-art and well-known ensemble methods:

Bagging its performance degrades when deals with stable base learners [34];

Boosting its highly sensitive to noise and outliers;

Random Forest its difficult to analyse, overfit noise data, it cannot predict beyond the range of the training data.

Moreover, the authors indicate that the performance of the analysed methods varies consistently because of the characteristics of the features in the datasets and which set of the features the learners are given.

Several authors have shown that DS obtain high performances in terms of accuracy on low dimensional datasets when compared to static ensemble methods and single classifiers [4, 5]. The main advantage of DS methods is the fact that, in theory, the DS methods have the capability of selecting the most competent classifiers in each region of the feature space, since these regions may have different classification difficulties, which would be unfeasible to achieve using a pre-defined (static) selection of classifiers. Nevertheless, many authors observed that DS techniques are still far from the upper bound performance of the oracle, which always predicts the correct label if at least one classifier in the ensemble predicts the correct label. Along with its high computation cost, the complexity of selecting the best classifier for a specific region in the feature space is perhaps the two most important drawbacks of DS methods. Cruz *et al.* [5] compared the DS methods with a k-NN and indicated that the later should be used for the classification of instances associated with a low degree of instance hardness, while DS methods were able to select the correct label on instances with a higher degree of instance hardness (samples that are located close to the decision border). Therefore, authors have therefore proposed solutions to improve the quality of the region of competence in low-dimensional datasets to increase classifier performance [5, 35]. . Over the past decade, DS techniques have been evaluated on low dimensional datasets and, to the best of our knowledge, there is no work in the literature that verifies the performance of the state-of-art DS methods for high-dimensional, small-instances datasets. Moreover, there is no adaptation of DS to adequately handle those datasets.

As stated by Cruz *et al.* (2017) [4], improvement on how the "region of competence" is defined is still an open problem in the area. Most DS methods use a k-NN as the method to define the "region of competence". As a consequence, when dealing with datasets datasets, the quality of the "region of competence" can be compromised, decreasing the performance of DS methods.

In datasets datasets, the task of selecting relevant features is one of the most important problems in machine learning. In two reviews Bólon-Canedo *et al.* [16, 36] reported the benefits of applying feature selection methods to improve classification, and highlighted the fact that feature selection methods are considered a *de facto* standard in machine learning and data analysis since their introduction. Tsymbal *et al.* (2001) [37] and Pechenizkiy *et al.* (2007) [38] demonstrated the benefits of integrating feature selection methods to the DS framework, to remove redundant and noisy features that will impact in the definition of the region of competence. However, the datasets used had a sample-feature ratio higher than one.

Cluster analysis is the task of grouping a set of samples based on some similarities of their features, i.e., data points in one group (cluster) are more similar to the points in the same group and dissimilar to data points in other groups. k-NN is one of the main strategies for distance-based grouping described in the literature and a core method in many DS frameworks. Traditional clustering algorithms consider all the dimensions of the dataset to learn about each sample and compute the similarities between samples [10]. In datasets datasets, many of the dimensions are irrelevant and can directly impact in the quality of the clusters retrieved [10, 39]. In addition, due to the properties of high-dimensional spaces (distance concentration and hubness), defining distance between data points in high-dimensional data is very difficult [10, 18–20, 40, 41]. Feature selection and extraction methods have been employed to remove irrelevant features to improve cluster quality [10, 39]. However, in high-dimensional data, a phenomena called *local feature relevance* happens, i.e., different subsets of features are relevant for different clusters [39]. Therefore, traditional feature selection and extraction methods which use all data points to determine the importance of each feature might not be suitable. Instead, subspace clustering methods localise their search and are able to elucidate clusters that exist in multiple, possible overlapping subspaces of features and/or samples [10].

This thesis, therefore, proposes a framework for dynamic selection (DS) methods with the incorporation of subspace clustering, named Subspace-Based Dynamic Selection (SBDS). We investigate whether it is possible to increase the performance in terms of accuracy of DS methods and improve knowledge discovery in high-dimensional small-instance datasets. To accomplish this, we use the main characteristics of the DS framework and integrate them with subspace clustering. Despite the large number of papers published on DS, there is no comprehensive study available verifying the use of DS and subspace clustering on high-dimensional datasets, also focusing on understanding feature importance on sample sets. Therefore, the SBDS and the cSBDS methods proposed here have an advantage of using a subspace clustering method to determine the best sets of features for each region of the feature space when comparing with RSS, giving meaning to the selection of the subspaces and increasing the changes of understanding how the feature within a subspace relate to each other, and why they were selected.

1.3 Aim and Objectives

The overall aim of this thesis is to overcome the limitations of the current DS methods by proposing a framework. This framework incorporates subspace clustering to determine the "regions of competence" into the DS framework, thereby overcoming existing issues with high-dimensional small-instance datasets when using distance-based approaches. The objectives are as follows:

1. Chapter 3 evaluates the performance of DS methods in synthetic and real world high-dimensional small-instance datasets before and after different feature selection approaches, and compares them with traditional ensemble methods (e.g. majority voting) and single classifiers;

2. Chapter 4 proposes two frameworks which combines the advantages of DS and subspace clustering methods to replace k-NN as the method to define the region of competence, and test the proposed frameworks on high-dimensional small-instance datasets, comparing the results with DS methods;

3. Chapter 5 investigates how subspace clustering selects the most important features and compares them with traditional feature importance approaches;

1.4 Research Questions

The above aim and objectives relate to three specific research questions to be asked in this thesis:

1. **What are the strengths and limitations of DS methods in high-dimensional small-instance datasets?** Whilst DS methods have been shown to perform statistically better than traditional MCS methods and single classifiers on datasets with more samples than features, there is no study available in the literature, to the best of our knowledge, that evaluates the performance of DS methods in high-dimensional small-instance datasets. In addition, an investigation is required to better understand the specific challenges and problems of small instance high-dimensional datasets.

2. **How can we incorporate subspace clustering methods to the DS framework?** Two frameworks were proposed both of them integrating a subspace clustering method into the traditional DS framework. The subspace clustering substitutes the region of competence steps in DS by selecting the most interesting subspaces that will be used as "regions" to evaluate the classifiers.

3. **How can we extract information in terms of feature importance for classification for each unseen sample?** By using subspace clustering methods it is possible to understand which are the most importance features to classify each sample. Feature importance techniques were also used to compared the results and evaluate the features found.

1.5 Thesis Outline

This section outlines each chapter of this dissertation.

Chapter 2 provide the literature review. It first gives a broad overview on MCS and DS methods. Next, it discusses the issues of high-dimensional datasets and methods that can improve the classification: feature selection and subspace clustering. Finally, the chapter discusses the concepts of the nearest subspace search which is one of the main concepts necessary for the understanding of our proposed SBDS framework.

Chapter 3 introduces the challenges of high-dimensional small instance datasets used in this dissertation. This serves as an evaluation of two criteria (hubness and distance concentration). In addition, we demonstrate how traditional dimensionality reduction methods perform on those datasets. Next, the chapter evaluates the DS methods in synthetic and real-world datasets with feature selection methods such as wrapper and filter methods.

Chapter 4 introduces the Subspace-Based Dynamic Selection (SBDS) framework. This is our proposed method that includes a subspace clustering method instead of the traditional k-NN method to define the region of competence. The chapter describes the two versions of the SBDS framework. The first one presented in Maciel-Guerra *et al.* (2020) [42] uses a distance metric to find the nearest subspace for each unseen sample. While the second version introduces a filter for the selected subspace clusters to improve performance and uses the performance of a Radial Basis Function Support Vector Machine (RBF-SVM) classifier to select the best subspaces, whilst also changing the way the subspaces are found. Both frameworks are validated on small instance high-dimensional datasets and they are compared to the traditional DS methods and majority voting.

Chapter 5 gives an overview of the features selected by the SBDS framework and analyse them by comparing with different feature importance methods. It also presents a different way to find the subspaces.

The thesis concludes with Chapter 6, where a summary of the contributions is provided. This also includes a discussion into how well the aim and objectives are met, as well as ideas for possible future directions of this research.

1.5.1 Contributions to knowledge

The research described in this thesis has demonstrated the applicability of different methods to understand and overcome the issues associated with small instance high-dimensional datasets. This work also highlights a core limitation of DS methods when applied to high-dimensional datasets due to their use of k-NN as the method to define to region of competence. This achieves one of the key objectives of the thesis: the evaluation of DS methods on small instance high-dimensional datasets. A key finding is that the DS methods are statistically equivalent to single classifiers and majority voting when compared to small instance high-dimensional datasets before and after feature selection approaches.

The most significant contribution of the thesis is the Subspace-Based Dynamic Selection (SBDS) framework, which achieves the key objective of a framework which contains the advantages of DS and subspace clustering methods. The two versions of the proposed SBDS framework were able to perform statistically better when compared to majority voting and single classifiers on real-word and synthetic data. This framework can be briefly outlined as follows:

- Find one-dimensional clusters using either a Gaussian Kernel Density Estimator (GKDE) or a Jenks Natural Break Points (NBP) approach

- After finding all one-dimensional clusters, a merge process is conducted to obtain the subspace clusters.

- For each unknown test sample, determine the nearest or best subspaces to train a classifier to predict the unseen sample.

In addition, the SBDS framework has the advantage of indicating which were the most important features for the classification, since for each unseen samples a potential different group of subspaces can be selected. The Classifier SBDS (cSBDS) framework was compared with different feature importance methods on synthetically created data and it indicates that on datasets with multiple clusters per class the cSBDS has the capability of uncovering the most important features better than traditional methods.

1.6 Academic Publications

The following publications were produced as a direct result of the work undertaken during the course of conducting this research:

1. **A. Maciel-Guerra, G. P. Figueredo, F. J. V. Zuben, E. Marti, J. Twycross, and M. J. C. Alcocer, "Microarray feature selection and dynamic selection of classifiers for early detection of insect bite hypersensitivity in horses," in *IEEE Congress on Evolutionary Computation, CEC 2019, Wellington, New Zealand, June 10-13, 2019*, IEEE, 2019, pp. 1157–1164. DOI:** 10.1109/CEC.2019.8790319 In this publication, we investigate the potential use of DS methods to classify protein microarray data, with a case study of equine insect bite hypersensitivity (IBH) disease. To the best of our knowledge DS was not previously applied to these data types. However, since most microarrays datasets have a low number of samples, we hypothesise that DS models will produce satisfactory results due to their ability to perform better when compared to traditional ensemble techniques for similar data. Results from traditional classifiers are compared to 21 different DS methods before and after performing feature selection. Our results indicate that DS methods do not outperform single and static classifiers on this high-dimensional dataset and their performance also does not improved after feature selection. Detailed description is presented as part of Chapter 3.

2. **A. Maciel-Guerra, G. P. Figueredo, and J. Twycross, "Dynamic selection**

of classifiers applied to high-dimensional small-instance data sets: Problems and challenges," in *LOD2020 - The Sixth International Conference on Machine Learning, Optimization, and Data Science – July 19-23, 2020 – Certosa di Pontignano, Siena – Tuscany, Italy*, Lecture Notes in Computer Science - LNCS, 2020. DOI: 10.1007/978-3-030-64583-0_56 In this publication, DS is employed on small instance high-dimensional datasets and a feature selection method is used to verify if the performance of the DS methods can be improved. Therefore, the performance of 21 DS methods was statistically compared against the performance of majority voting on 10 high-dimensional datasets and with a filter feature selection method. We found that majority voting is among the best ranked classifiers and none of the DS methods perform statistically better than it with and without feature selection. Moreover, we demonstrated that feature selection does improve the performance of DS methods. Detailed analysis is presented as part of Chapter 3.

3. **A. Maciel-Guerra, G. P. Figueredo, F. J. V. Zuben, E. Marti, J. Twycross, and M. J. C. Alcocer, "Subspace-based dynamic selection: A proof of concept using protein microarray data," in *WCCI - World Congress on Computational Intelligence, The International Joint Conference on Neural Networks (IJCNN) 2020, Glasgow, UK, July 19-24, 2020*, IEEE, 2020. DOI:** 10.1109/IJCNN48605.2020.9207611 In this paper, we propose a two-stage framework based on subspace clustering using a Gaussian Based Estimator, followed by a k-Nearest subspace search mechanism to overcome these limitations of dynamic selection. The idea of subspace allows for regions of competence with different numbers of instances and dimension sizes. Our hypothesis is that by using our framework, we will achieve comparable results to the state-of-the-art dynamic selection, with the benefit of producing a model that helps to understand the importance of sets of features for the patterns found within the data. We test our approach to a high-dimensional microarray data of insect bite hypersensitivity in horses. Results show that our approach is comparable to traditional dynamic selection methods in

terms of accuracy. In addition, it facilitates the interpretability of the feature importance for each class of the dataset. Detailed description is presented in Chapter 4.

4. **A. Maciel-Guerra, G. P. Figueredo, and J. Twycross, "Classifier subspace-based dynamic selection for high-dimensional data,"** ***In Review*****, 2022** In this paper, we propose the classifier subspace-based dynamic selection (cSBDS) framework which incorporates a subspace clustering method, using a Gaussian kernel density estimator, to the dynamic selection framework. A RBF-SVM classifier is used to select the most important subspaces and give the final prediction. The subspace clustering approach separates the high-dimensional feature space into small feature spaces with a reduced number of features and samples in each one. Our hypothesis is that the cSBDS framework can perform statistically better when compared to the state-of-art DS methods. To test this hypothesis, the cSBDS was evaluated on ten small instance high-dimensional datasets and results indicate that our approach performs statistically better when compared to the SBDS framework, 10 DS methods and the majority voting technique. Detailed description is presented in Chapter 4.

Chapter 2

Literature Review

***

2.1 Introduction

Data mining is the study of discovering insightful, interesting and novel patterns, as well as descriptive, understandable and predictive models from large-scale data. From an analytical perspective, data mining is challenging because of the amount of data types, problems and methods encountered. Nonetheless, data mining can be often connected to four different categories: association pattern mining, clustering, classification and outlier detection [1, 2]

Consider a multidimensional dataset D with n records, and d attributes. Such a dataset D may be represented as an $n \times d$ matrix D, in which each row corresponds to one record and each column corresponds to a dimension [1, 2]. The relationship between data items are one of two kinds:

1. **Relationship between the columns:** many data mining problems are directed toward a specific goal that is sometimes represented by the value of a particular feature in the data. This particular feature is referred to as the class label from which the relationships of the remaining features in the data with respect to this special feature are learned. Therefore, these methods are called supervised learning. The data used to learn these relationships is referred to as the training data. The learned model may then be used to determine the estimated class labels for records, where the label is unknown [1, 2].

2. **Relationship between rows:** the goal is to determine a set of rows, in which the values in the corresponding columns are related. Clustering methods are considered unsupervised and one of the possible definitions is the following: given a data matrix D, partition its rows (records) into sets $C_1 \dots C_k$, such that the rows (records) in each cluster are "similar" to one another [1, 2].

Machine learning and pattern recognition methods are frequently used to solve problems in several areas like life sciences, medicine, recognition systems, data streams and others [1, 2]. A recurrent task in machine learning is the choice of classifier to solve a problem, of which is expected to choose a classifier that better generalises the problem and has the highest recognition rate [44]. However, this is not a trivial task, and the "no free lunch" theorem states that there is no classifier that performs better across all problems [44]. Several studies, nonetheless, have demonstrated that ensemble methods usually perform better that single classifiers [3–7, 28, 33], because several weak classifiers can be grouped together to form a better learner.

One of the most important tasks surrounding algorithms of ensemble of classifiers is the decision of which classifier or classifiers will be selected. Therefore, from a pool of different classifiers an algorithm must select a group that can achieve optimum recognition rates. It is possible to perform this using two different methods: (1) Static Selection (SS); (2) Dynamic Selection (DS). The first selects a group of classifiers for all test patterns, while the second select a single or a group of classifiers for each different test pattern. DS methods have been chosen more often recently over static methods due to their capacity of creating different classifier configurations according to each test pattern. It is reasonable to assume that, in many cases, different test patterns will be associated with different classification difficulties. Therefore, the ability to choose a group of classifiers for each test pattern gives us reasons to believe that DS methods can overcome static selection methods [4, 6, 7].

This chapter reviews the literature related to the project. Firstly, a brief description of multiple classifier systems (MCS) is given to introduce the basic concepts of Dynamic Selection (DS). Next, challenges and opportunities of studying high-dimensional data are discussed. The following section introduces technical details on how to perform dimensionality reduction. Finally, methods of subspace clustering and nearest subspace search are discussed to provide the necessary concepts for the understanding of the Subspace-Based Dynamic Selection (SBDS) framework proposed.

2.2 Multiple Classifier Systems

Empirical studies have demonstrated that a multiple classifier systems (MCS) might increase efficiency and classification accuracy when compared with a single classifier in pattern recognition and forecasting problems. Nowadays, they are widely used to solve real-world problems, such as intrusion detection, recommendation systems, face recognition and time series predictions [4, 5, 12, 45–48]. The idea is that a combination of "different" classifiers might have a strong degree of "independence", *i.e.* the prediction errors committed by a classifier c_i can be overcome by the correct classification of other classifiers. Initially, the MCS methods had their final prediction given by the combination of the prediction of the different classifiers used. In 1994, Ho *et al.* [49] proposed a selection approach to choose a single or a group of classifiers that were more suitable for undertaking a given classification task, instead of combining the outputs of all classifiers in the MCS.

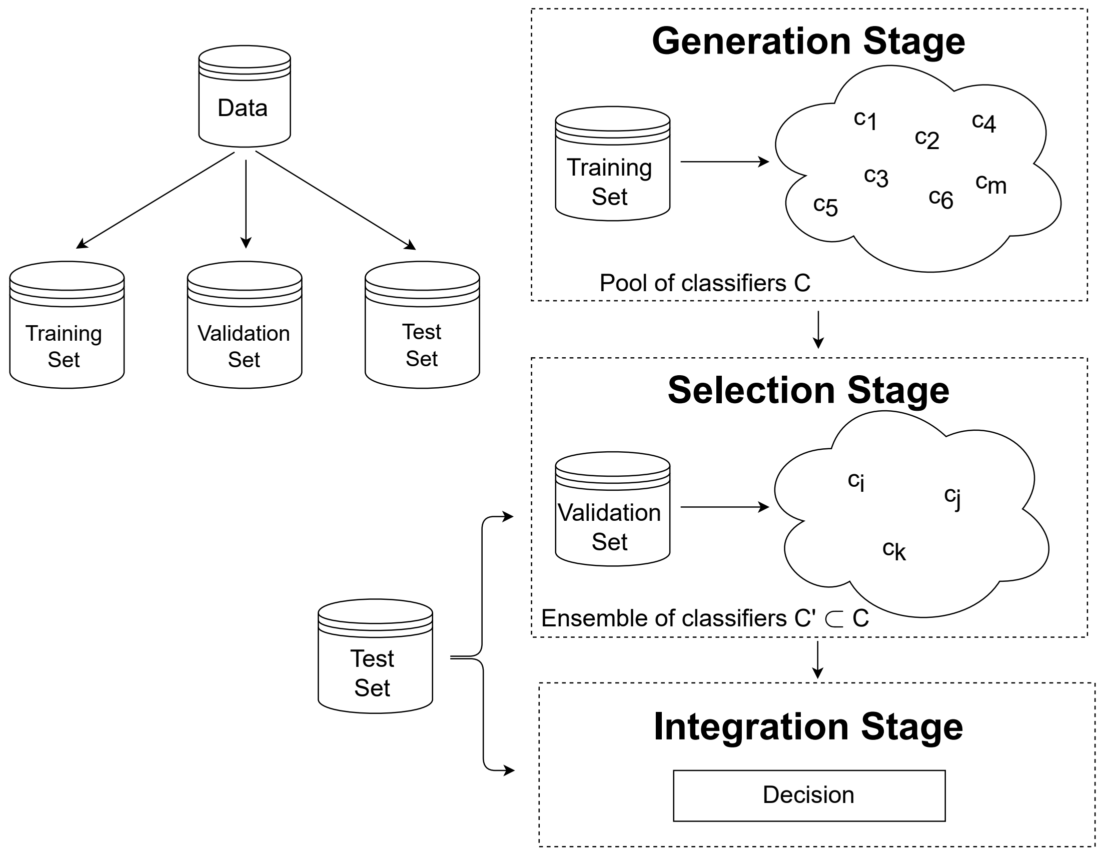


Figure 2.1: Basic MCS framework showing the three stages: generation, selection and integration.

The basic framework of an MCS is presented on Figure 2.1 and the most relevant techniques used in each stage is presented on Figure 2.2. MCS are essentially composed of three major stages: (1) **pool generation**, (2) **selection** and (3) **integration**. On the **pool generation stage**, the main goal is to train a pool of classifiers that are both accurate and diverse. On the **second stage**, based on the pool of classifiers, the goal is to select a single or an ensemble of classifiers from the pool of classifiers. This stage can be divided into two groups: static and dynamic selection. In the first group, the classifiers are selected during the training stage and are fixed for all the unknown test samples, while the later selects a different set of classifiers for each test sample. The **final stage** consists of combining the outputs of the selected ensemble of classifiers according to a combination rule [4–6]. In addition, Cruz *et al.* [4] presented a taxonomy for a multiple classifier system considering the main approaches proposed in for each stage (Figure 2.2), as further detailed next.

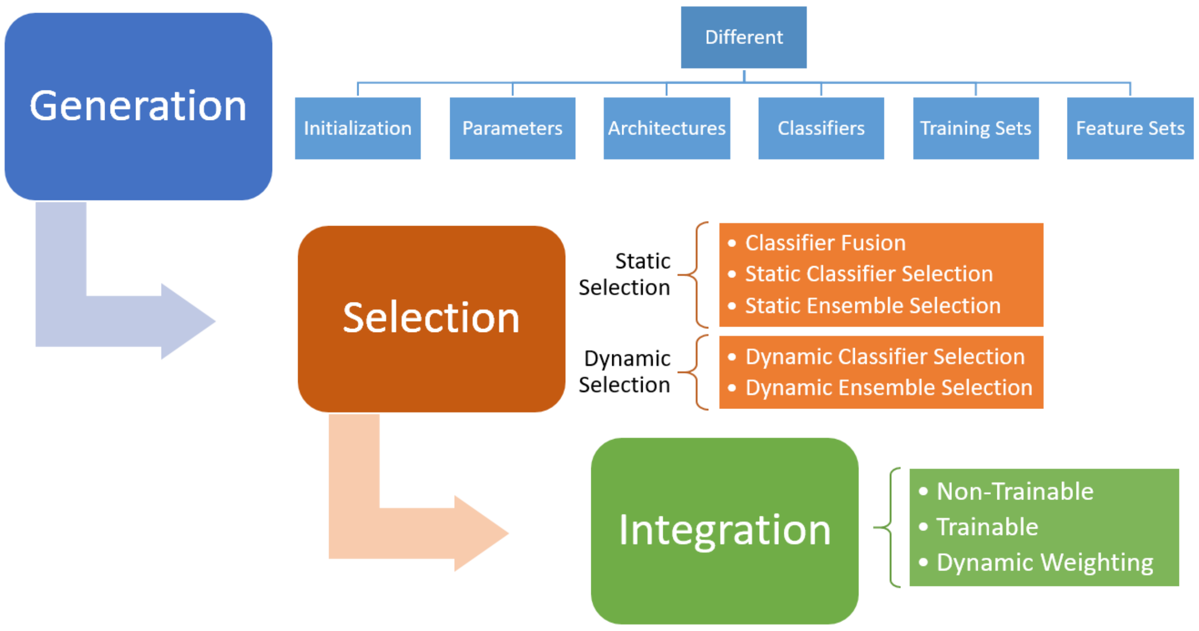


Figure 2.2: Main approaches for each of the three stages of multiple classifier systems (adapted from [4, 50]).

2.2.1 Generation

In the first stage, the main goal is to train a pool of classifiers that are diverse and accurate, *i.e.* the classifiers must have an error rate lower than random guessing (accurate) and two classifiers must make different errors on new samples (diverse). Their combination, therefore, is likely to improve the classification accuracy [4, 6, 51]. Figure 2.2 shows six main strategies to generate a pool of classifiers: different (1) initialisation, (2) parameters, (3) architectures, (4) classifiers, (5) training sets and (6) feature sets, that can be used separately or combined [4].

2.2.2 Selection

Static and dynamic classifier ensembles are the two main methodologies of the second stage of a MCS. Static selection means that the ensemble of classifiers is selected during the training phase according to a selection criteria estimated on the validation set. The optimal solution found is fixed and used for the classification of all unknown test samples. On the other hand, DS means that a single or an ensemble of classifiers are selected on the testing stage according to each unknown test sample and could be different between the different test samples [4, 51, 52]. Figure 2.3 shows the difference between static and dynamic selection.

2.2.2.1 Static Selection

In general, there are three approaches for static selection: classifier fusion, static classifier selection and static ensemble selection. The first has higher chances of having inaccurate and redundant classifiers [50, 52]. Therefore, the other two have recently received more attention .

Classifier Fusion

Simple or weighted majority voting, negative correlation learning, adaboost, bagging, boosting and random forest are the most common decision making methods employed on classifier fusion [50].

Static Selection

This method first generates the local region of competence in the feature space during the training phase by using either the training set itself or an independently validation set. Next, it selects a single best or an ensemble of classifiers for each region which will be fixed for all test samples. Finally, each test sample is classified according to the region it belongs using the classifier from that region [50].

2.2.2.2 Dynamic Selection of Classifiers

Differently from static selection, the selection of the classifiers is done during the selection stage of the MCS [4, 6, 48, 52]. In other words, a single (dynamic classifier selection) or an ensemble (dynamic ensemble selection) is selected specifically to classify each new test sample. As a rule, the dynamic classifier ensemble has three basic steps (Figure 2.4). First, the system needs to generate the local region of competence based on the training set on an independent validation set. Second, the system uses a selection criterion to dynamically determine the competence level of the classifiers. The selection is dynamic because generally these two steps are performed during the testing phase.

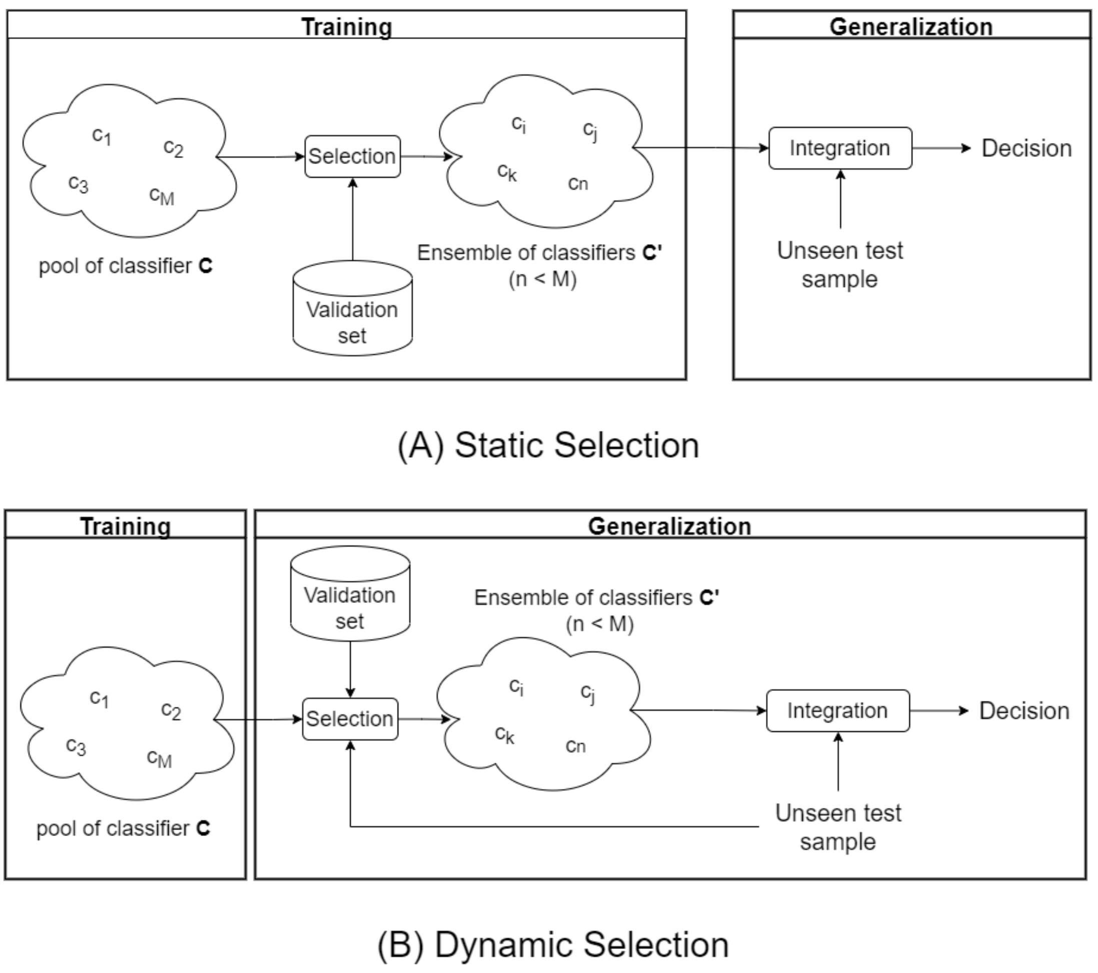


Figure 2.3: Differences between static ensemble selection and dynamic selection [4].

However, there are some approaches in dynamic classifier ensemble that predetermine the region of competence during the training phase and only select the best classifiers during the testing phase [4, 6, 48].

2.2.3 Integration

The final stage consists of combining the output of the selected classifiers according to a combination rule. There are three strategies for this stage: non-trainable, trainable and dynamic weighting.

2.2.4 Limitations of multiple classifier systems

The ultimate goal of supervised learning for classification is to correctly predict the class of a sample based on previous knowledge from existing data. Many algorithms

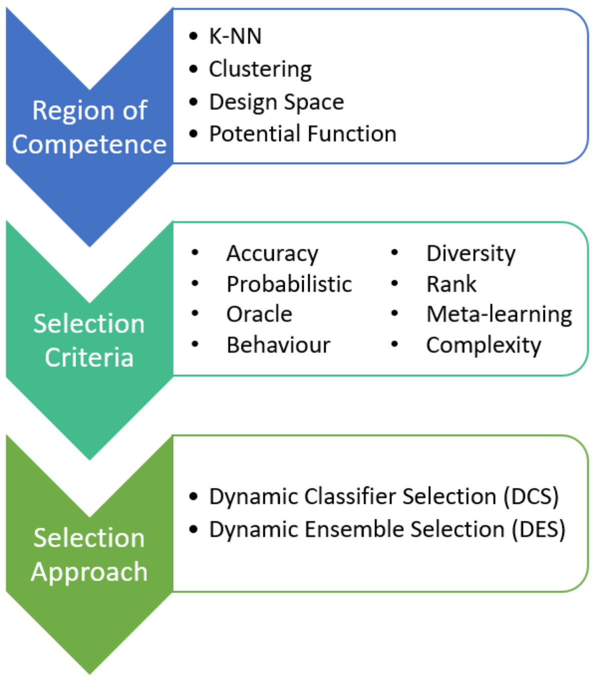


Figure 2.4: Main approaches considering the three different steps of DS (adapted from [4]).

have been proposed such as random forest [53], neural networks [54, 55], and support vector machines [56, 57]. However, the increase in data dimensionality causes issues with regarding scalability and performance. Moreover, the classification ability of a single classifier could be limited [33]. In 2015, Meshram and Shinde [28] presented a survey on ensemble methods for high-dimensional datasets. The authors show that individual classifiers are usually not able to handle the noise and imbalance data in high-dimensional datasets [28, 33]. On the other hand, ensemble classifiers provide a better classification performance [28]. Moreover, the authors indicate that the performance of the analysed methods varies considerably because of the characteristics of the features in the datasets and which set of the features the learners are presented with.

The following section describes DS methods while a comparison between the different methods is given. Appendix A gives an in depth description of the individual DS methods used in this thesis.

2.3 Dynamic Selection of Classifiers

Recently, DS methods have been preferred over static methods due to their ability to create different classifier configurations, i.e. different groups of classifiers are experts in different local regions of the feature space. As for many cases, different samples are associated with different classification difficulties and the ability to choose a group of classifiers can possibly overcome static selection methods limitations [4, 6, 7].

For DS methods to achieve optimal recognition rates they need to select the most competent classifiers for any given test sample, which can be done by measuring different selection criteria depending on the technique used (accuracy, ranking, behaviour, diversity, probabilistic, complexity and meta-learning) [4, 6]. A local region of the feature space surrounding the test sample (Region of Competence) is used to estimate the competence of each classifier according to any selection criteria.

Table 2.1 shows the most important DS methods found in the literature in terms of their selection criteria. These methods were chosen due to their differences in the selection criteria and because they state-of-art methods. The majority of DS techniques relies on k-Nearest Neighbour (k-NN) algorithms (Table 2.1) and the quality of the neighbourhood can have a huge impact on the performance of DS methods [4–6]. For DS methods using k-NN, as indicated in the Table 2.1, the region of competence size is k; for approaches not adopting k-NN, all data is used to make the prediction. For the methods using a k-NN, the "region of competence" is defined as the k closest samples to the unknown test sample.

2.3.1 Pool generation and diversity

Intuition says that a good set of base classifiers should have methods with high recognition rates and be as diverse as possible. Diversity is not an exact concept and defining it is not trivial [31, 32]. However, classifiers that make statistically different errors are a good starting point to create a pool and improve the system's recognition rate. If knowledge about the diversity of a set of classifiers is good, this makes it possible to choose

Table 2.1: DS methods investigated

Name	Selection criteria	DS Method	Region of Competence	Reference
Classifier Rank (CR)	Ranking	DCS	k-NN	[58]
Modified Classifier Rank (MCR)	Ranking	DCS	k-NN	[59]
Overall Local Accuracy (OLA)	Accuracy	DCS	k-NN	[59]
Local Class Accuracy (LCA)	Accuracy	DCS	k-NN	[59]
A Priori	Probabilistic	DCS	k-NN	[60]
A Posteriori	Probabilistic	DCS	k-NN	[60]
Multiple Classifier Behaviour (MCB)	Behaviour	DCS	k-NN	[61]
Modified Local Accuracy (MLA)	Accuracy	DCS	k-NN	[62]
DES - k-Means (DES-kMeans)	Accuracy & Diversity	DES	k-Means	[63, 64]
DES - k-Nearest Neighbour (DES-kNN)	Accuracy & Diveristy	DES	k-NN	[63, 64]
KNORA - Eliminate (KNORA-E)	Oracle	DES	k-NN	[7]
KNORA - Union (KNORA-U)	Oracle	DES	k-NN	[7]
DES - Exponential (DES-EXP)	Probabilistic	DES	All training samples	[65]
DES - Randomised Reference Classifier (DES-RRC)	Probabilistic	DES	All training samples	[29]
DES - Minimal Difference (DES-MD)	Probabilistic	DES	All training samples	[66]
DES - Kullback-Leibler Divergence (DES-KL)	Probabilistic	DES	All training samples	[67]
DES - Performance (DES-P)	Probabilistic	DES	All training samples	[67]
KNOP - Eliminate (KNOP-E)	Behaviour	DES	k-NN	[68]
KNOP - Union (KNOP-U)	Behaviour	DES	k-NN	[68]
Meta-Learning - DES (Meta-DES)	Meta-learning	DES	k-NN	[30]
Dynamic Selection on Complexity (DSOC)	Accuracy & Complexity	DCS	k-NN	[69]

the fusion method better [44]. The link between how diversity measures are related to the recognition rate of a classifier set is still a matter of investigation. However, experimental studies have observed that: the greater the diversity, the greater the performance of MCS methods. They also showed that the properties of a set of classifiers that are desirable (high recognition rate and diversity) to obtain a successful combination are not common in practice [44].

The different classifier generation approaches aim to create a good pool of classifiers (with high hit rate and diversity), in order to have a better performance after their combination. Some of the more popular methods are seen below:

1. **Bagging**: an acronym for Bootstrap AGGregaTING [34], is one of the first methods for generating ensemble of classifiers proposed, one of the most intuitive, and perhaps one of the simplest algorithms for generating a set of classifiers, with a good performance. Bagging is based on the idea that bootstrap samples from the training set will show a small change with respect to the original set, but enough difference to produce diverse classifiers. Thus, for bootstrapping the different subsets of training data are randomly drawn - with replacement - from the entire training dataset. Then, each subset of training data is used to train a different base classifier. Finally, the individual classifiers are combined by averaging or majority voting

of their decisions. Although bagging is a good algorithm, as the diversity of the classifiers is obtained using several bootstrap "replicates" of the training set, it is only effective when a small change in the set can cause a significant change in the model. Therefore, to make use of training set variations, the base classifier must be unstable or non-linear, that is, small changes in the training set must lead to large changes in the classifier's output. Otherwise, the resulting set will be a collection of nearly identical classifiers, therefore unlikely to improve the performance of a single classifier. Examples of unstable classifiers are neural networks and decision trees, while k-NN is an example of a stable classifier [4, 7, 35, 67].

2. **Boosting:** similar to bagging, boosting [70] also creates an ensemble of classifiers by sample subsets of the training data and combines the output by majority voting. However, in boosting, sampling is strategically oriented to provide the most informative training data for each classifier consecutively. In essence, each iteration of boosting creates three weak classifiers: the first classifier c_1 is trained with a random subset of the training data. The training data subset for the second classifier c_2 is chosen as the most informative subset, given c_1. More specifically, c_2 is trained on the training data only half of which is correctly classified by c_1, and the other half of which is miss-classified. The third classifier c_3 is trained with instances where c_1 and c_2 disagree. The three classifiers are combined through majority voting [4, 35, 71].

3. **Random Subspace Sampling (RSS):** one way to improve diversity in a pool of classifiers is to train the individual classifiers with different subsets of the feature space. The selection of features aims at a more efficient computation and a higher recognition rate of the set. RSS [72] is a method of selecting subsets of random features for the construction of a pool of classifiers. It is similar to bagging but instead of sampling instances, it samples features without repetition, as it would be pointless to include a feature more than once. RSS randomly selects an arbitrary number of feature subspaces from the original space, and builds each classifier in each subspace. This randomisation should create classifiers that are complementary

and so the combination can be done by simple fusion rules [4, 7, 35].

Most of the studies in the area of DS have focused on finding the neighbourhood of a new data instance and choosing the most competent classifier(s) to make the prediction. According to Yasar *et al.* [73] the focus on having a diverse ensemble of classifiers was ignored on previous studies because the DS systems is not required to generalise to all samples, i.e. since the system focus on prediction the class of a single instance at each time, the classifiers' performance over the different regions is not relevant. The majority of the methods in DS uses a bagging approach to generate the pool of classifiers, with the base learner being the difference across different studies. Also, to date, a few dynamic selection models have incorporated diversity with other competence measures to perform an ensemble selection. The two methods that do consider it (DES-kMeans and DES-kNN) usually do not present a good overall performance [4].

It is worth noticing that diversity in dynamic selection methods might not be that important when compared to static selection methods. Because after selecting the classifiers from the pool, the goal should be to achieve a consensus between the classifiers for a specific region of the feature space. In addition, Cruz *et al.* [4] indicated that increasing the diversity at instance level may not improve the performance of DS methods. Nonetheless, Yasar *et al.* [73] proposed that diversity still plays a role in DS prediction, because in the ideal world the classifiers chosen in DS should behave similarly in the region of the feature space from which their were selected, but differently outside this area. Therefore, they should have a high local accuracy, with a low local diversity but a high diversity in other regions.

2.3.2 Comparison between Dynamic Selection (DS), Static Selection (SS) and single classifiers

Table 2.2 presents the main results from the literature. It shows the papers where these methods were cited and the datasets on which they were tested. Moreover, we separated each method in terms of their selection criteria and the type of region of competence.

Table 2.2: Summary of the main features of the DS methods and their performance in the literature.

Ref.	Method	Category	Sel. Type	RofC Type	Eval in	Pool Type (size)	Datasets	SB	CC	SS	Other DS
[59]	CR	Ranking	DCS	k-NN	[59]	Het(5)	3S	(1,0,2)	(2,0,1)	NA	(1,0,8)
-	-	-	-	-	[74]	Hom(100)	28S/2L	(16,0,14)	(25,0,35)	(13,0,17)	(99,0,171)
[59]	MCR	Ranking	DCS	k-NN	[59]	Het(5)	3S	(2,0,1)	(2,0,1)	NA	(5,0,4)
[59]	OLA	Accuracy	DCS	k-NN	[59]	Het(5)	3S	(2,0,1)	(3,0,0)	NA	(4,0,5)
-	-	-	-	-	[60]	Het(5)	3S	(3,0,0)	(0,0,6)	NA	(4,1,4)
-	-	-	-	-	[62]	Het(3)/Het(4)	1S/1L	(2,0,0)	(3,0,0)	NA	(2,0,0)
-	-	-	-	-	[7]	Hom(10)	5S/1L	(27,6,21)	(36,3,15)	NA	(122,50,98)
-	-	-	-	-	[30]	Hom(100)	27S/3L	(18,0,12)	(27,2,31)	(13,0,17)	(80,36,94)
-	-	-	-	-	[69]	Hom(100)	29S/1L	(22,1,7)	(20,0,10)	NA	(99,3,48)
-	-	-	-	-	[74]	Hom(100)	28S/2L	(17,0,13)	(25,1,34)	(13,0,17)	(99,33,138)
[59]	LCA	Accuracy	DCS	k-NN	[59]	Het(5)	3S	(3,0,0)	(3,0,0)	NA	(8,0,1)
-	-	-	-	-	[60]	Het(5)	3S	(3,0,0)	(1,0,5)	NA	(3,0,6)
-	-	-	-	-	[66]	Het(9)	6S	(3,0,3)	(2,0,4)	NA	(10,0,2)
-	-	-	-	-	[29]	Het(10)/Hom(50)	22S	(17,0,27)	(30,0,58)	NA	(48,7,121)
-	-	-	-	-	[67]	Het(11)/Hom(11)	14S	(16,0,12)	(4,0,24)	NA	(26,2,112)
-	-	-	-	-	[7]	Hom(10)	5S/1L	(20,8,26)	(27,4,23)	NA	(122,35,113)
-	-	-	-	-	[30]	Hom(100)	27S/3L	(18,0,12)	(29,1,30)	(13,0,17)	(81,32,97)
-	-	-	-	-	[69]	Hom(100)	29S/1L	(23,1,6)	(21,0,9)	NA	(116,4,30)
-	-	-	-	-	[74]	Hom(100)	28S/2L	(18,0,12)	(29,1,30)	(14,0,16)	(112,28,130)
[60]	a Priori	Probabilistic	DCS	k-NN	[60]	Het(5)	3S	(3,0,0)	(0,0,6)	NA	(1,1,7)
-	-	-	-	-	[7]	Hom(10)	5S/1L	(19,8,27)	(28,4,22)	NA	(104,38,128)
-	-	-	-	-	[69]	Hom(100)	29S/1L	(12,0,18)	(8,0,22)	NA	(41,1,108)
[60]	a Posteriori	Probabilistic	DCS	k-NN	[60]	Het(5)	3S	(3,0,0)	(2,0,4)	NA	(9,0,0)
-	-	-	-	-	[7]	Hom(10)	5S/1L	(16,7,31)	(23,4,27)	NA	(70,26,174)
[61]	MCB	Behaviour	DCS	k-NN	[61]	Het(3)	2S	(2,0,0)	(4,0,0)	NA	NA
-	-	-	-	-	[66]	Het(9)	6S	(4,0,2)	(3,0,3)	NA	(6,0,6)
-	-	-	-	-	[29]	Het(10)/Hom(50)	22S	(18,0,26)	(31,0,57)	NA	(49,5,122)
-	-	-	-	-	[67]	Het(11)/Hom(11)	14S	(15,0,13)	(6,1,21)	NA	(40,5,95)
-	-	-	-	-	[30]	Hom(100)	27S/3L	(19,0,11)	(34,0,26)	(17,2,11)	(67,28,115)
-	-	-	-	-	[74]	Hom(100)	28S/2L	(19,0,11)	(34,0,26)	(17,2,11)	(94,29,147)
[62]	MLA-Macleod	Accuracy	DCS	k-NN	[62]	Het(3)/Het(4)	1S/1L	(2,0,0)	(3,0,0)	NA	(2,0,0)
-	-	-	-	-	[29]	Het(10)/Hom(50)	22S	(20,0,24)	(31,0,57)	NA	(78,6,92)
-	-	-	-	-	[67]	Het(11)/Hom(11)	14S	(15,0,13)	(5,0,23)	NA	(28,5,107)
[62]	MLA-Euclidean	Accuracy	DCS	k-NN	[30]	Hom(100)	27S/3L	(11,0,19)	(22,0,38)	(11,0,19)	(59,11,140)
-	-	-	-	-	[74]	Hom(100)	28S/2L	(11,0,19)	(22,0,38)	(11,0,19)	(74,9,187)
[63]	DES-kNN	Accuracy/Diversity	DES	k-NN	[63]	Het(10)	2S	(2,0,0)	NA	(1,0,1)	(0,0,2)
[63]	DES-kMeans	Accuracy/Diversity	DES	k-Means	[63]	Het(10)	2S	(2,0,0)	NA	(2,0,0)	(2,0,0)
[65]	DES-EXP	Probabilistic	DES	all Val	[65]	Het(11)	5S	(4,0,1)	(15,0,0)	NA	NA
[66]	DES-MD	Probabilistic	DES	all Val	[66]	Het(9)	6S	(1,0,5)	(1,0,5)	NA	(2,0,10)
[29]	DES-RRC	Probabilistic	DES	all Val	[29]	Het(10)/Hom(50)	22S	(39,0,5)	(70,0,18)	NA	(170,0,6)
-	-	-	-	-	[4]	Hom(100)	29S/1L	NA	NA	NA	(89,1,60)
-	-	-	-	-	[74]	Hom(100)	28S/2L	(26,0,4)	(52,0,8)	(29,0,1)	(165,1,104)
[67]	DES-P	Accuracy	DES	k-NN	[67]	Het(11)/Hom(11)	14S	(22,1,5)	(22,2,4)	NA	(125,3,12)
-	-	-	-	-	[4]	Hom(100)	29S/1L	NA	NA	NA	(53,3,94)
[67]	DES-KL	Probabilistic	DES	all Val	[67]	Het(11)/Hom(11)	14S	(22,0,6)	(20,1,7)	NA	(122,0,18)
-	-	-	-	-	[4]	Hom(100)	29S/1L	NA	NA	NA	(50,2,98)
[7]	KNORA-E	Oracle	DES	k-NN	[7]	Hom(10)	5S/1L	(28,8,18)	(44,2,8)	NA	(155,41,74)
-	-	-	-	-	[29]	Het(10)/Hom(50)	22S	(25,1,18)	(28,1,59)	NA	(84,4,88)
-	-	-	-	-	[67]	Het(11)/Hom(11)	14S	(19,0,9)	(13,0,15)	NA	(71,1,68)
-	-	-	-	-	[30]	Hom(100)	27S/3L	(22,0,8)	(34,2,24)	(16,1,13)	(91,27,92)
-	-	-	-	-	[69]	Hom(100)	29S/1L	(9,1,20)	(4,4,22)	NA	(11,3,136)
-	-	-	-	-	[74]	Hom(100)	28S/2L	(22,0,8)	(34,2,24)	(17,1,12)	(118,27,125)
[7]	KNORA-U	Oracle	DES	k-NN	[7]	Hom(10)	5S/1L	(21,9,24)	(43,5,6)	NA	(120,44,106)
-	-	-	-	-	[30]	Hom(100)	27S/3L	(22,0,8)	(38,0,22)	(19,1,10)	(94,31,85)
-	-	-	-	-	[69]	Hom(100)	29S/1L	(16,1,13)	(11,1,18)	NA	(52,4,94)
-	-	-	-	-	[4]	Hom(100)	29S/1L	NA	NA	NA	(59,2,89)
-	-	-	-	-	[74]	Hom(100)	28S/2L	(22,0,8)	(38,0,22)	(19,1,10)	(127,31,112)
[68]	KNOP-U	Behaviour/Oracle	DES	Similarity	[68]	Hom(100)	4S/1L	(10,0,0)	(10,0,0)	NA	NA
-	-	-	-	-	[74]	Hom(100)	28S/2L	(18,0,12)	(37,0,23)	(17,2,11)	(136,25,109)
[68]	KNOP-E	Behaviour/Oracle	DES	Similarity	[30]	Hom(100)	27S/3L	(18,0,12)	(37,0,23)	(16,2,12)	(104,25,81)
[30]	Meta-DES	Meta-Learning	DES	k-NN	[30]	Hom(100)	27S/3L	(30,0,0)	(57,0,3)	(28,0,2)	(169,0,41)
-	-	-	-	-	[4]	Hom(100)	29S/1L	NA	NA	NA	(79,1,70)
[69]	DSOC	Complexity	DES	k-NN	[69]	Hom(100)	29S/1L	(28,0,2)	(23,0,7)	NA	(123,1,26)

In the table, the column Reference (Ref.) show who proposed them while the columns Method, Category, Selection Type (Sel. Type) and Region of Competence Type (RofC Type) indicate the most important characteristics of each method. The column Category highlights the different criteria of dynamic selection methods: ranking, accuracy, probabilistic, diversity, oracle, behaviour, meta-learning and complexity. The column Sel. Type separates the methods into two groups: dynamic classifier selection (DCS) and dynamic ensemble selection (DES). The column RofC Type presents the method to determine the region of competence: k-Nearest Neighbour (k-NN), k-Means, similarity, all Val (region of competence not defined, all validation samples are used).

The last 7 columns of table 2.2 present the papers where each method was evaluated; the type of pool of classifiers (Homogeneous or Heterogeneous) and its respective size; the number of datasets and their size (S = small and L = Large for datasets with more than 10.000 instances); while the last four columns present the number of wins, ties and losses of each method against the single best classifier in the pool (SB), all classifiers in the pool (CC), static selection methods (SS) and other dynamic selection method (other DS). The rationale behind the computation of wins, ties and losses is to have an approach that allows us to compare different papers, even if they used different datasets and/or different classifiers.

Figure 2.5 presents the comparison results of the methods from Table 2.2. Dynamic selection methods perform better when compared with SB, CC and SS. This indicates why DS have been preferred in the literature over static selection and traditional classifier fusion such as majority voting.

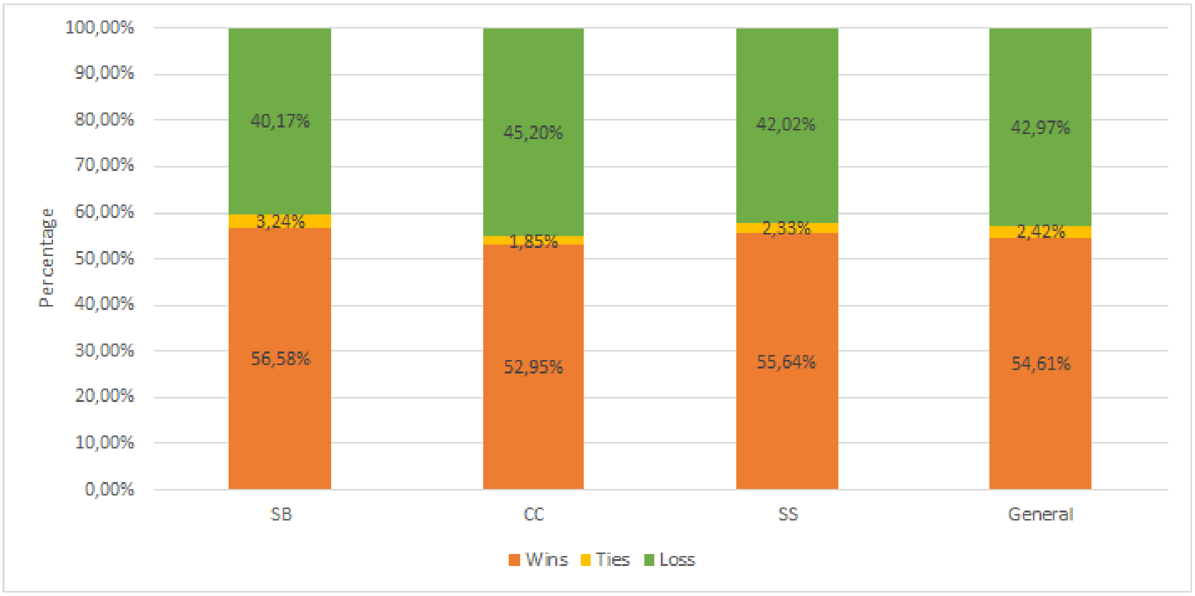


Figure 2.5: Performance of DS methods presented in table 2.2 in terms of percentage of wins, ties and losses when compared to SB, CC, SS and General (any alternative selection)

2.3.3 Limitations of dynamic selection methods

In the last two decades, a number of researchers have shown that for low-dimensional data, dynamic selection (DS) outperforms single robust classifiers and traditional combination methods, such as majority voting, bagging and boosting [4–7, 29, 30]. Nonetheless, for high-dimensional data, Maciel-Guerra *et al.* (2019) [8] and Maciel-Guerra *et al.* (2020) [9] showed that DS fails to perform as well, and the authors demonstrated that was due to use of the k-NN approach embedded in most DS methods. Also, according to them, DS methods performed statistically equivalent to majority voting when applied to high-dimensional datasets. Chapter 3 discusses in more details the results obtained and how they assisted in identifying the need for our thesis contributions. The following section, describes some challenges about high-dimensional datasets.

2.4 High-Dimensional Data

Exploring associations, making reliable predictions and extracting information are some of the problems that are yet to be solved in high-dimensional data [11, 15]. According to Verleysen and François (2005) [17], traditional machine learning techniques were often created having in mind intuitive properties and examples in low-dimensional datasets. However, when tackling high-dimensional data, collinearity, numerical instability, overfitting, model instability are some of the known problems that can occur. Moreover, high-dimensional spaces have geometrical properties that are not intuitive [17], for instance:

- *Distance concentration*: within very high-dimensional spaces, the distance between all data instances become almost equal, making, therefore, nearest neighbours to be unable to distinguish between “near” or “far” data points [20].

- *Hubness*: Let $D \subset \mathbb{R}^d$ be a set of d-dimensional points and $N_k(\vec{\mathbf{x}})$ the number of *k-occurrences* of each point $\vec{\mathbf{x}} \in D$, i.e., the number of times a point $\vec{\mathbf{x}}$ appears among the k-NN of all points in D, according to some distance metric [20]. According to Radovanovic *et al.* (2010) [20], as d increases, the distribution of N_k becomes considerably skewed to the right, resulting in the appearance of *hubs*, i.e., points that are “popular” nearest neighbours.

Therefore, these properties of high-dimensional spaces (distance concentration and hubness) can directly affect machine learning application, specially the ones that deal with distance metrics such as k-NN. Our hypothesis is that subspace clustering methods can overcome these issues, since they are able to select a subset of features and samples.

In general, the ensemble classifiers provide better classification accuracy than individual classifiers [28]. However, many ensemble methods proposed in the literature are do not perform accurately in high-dimensional biomedicine data, according to the recent survey made by Meshram & Shinde (2015) [28].

Chapter 3 will discuss in more details the issues regarding the ”curse of dimensionality” in the datasets studies in this research. It will also present the reasons of why DS methods fail to perform better when compared to traditional ensemble methods. The following section describes the traditional dimensionality reduction methods to overcome some of the issues of high-dimensional datasets. Feature selection methods are described and compared in terms of their advantages and disadvantages.

2.5 Feature Selection

Feature selection is the process of reducing the number of features in a dataset. In most of the times they are employed as a pre-processing step in pattern analysis to extract the useful information in the data [75]. This can be in terms of a better representation of the data or better discrimination of the classes [75]. This thesis focus on feature selection methods due to their simplicity and the possibility of analysing which were the most important features for the classification. Moreover, feature extraction/transformation methods transform the original feature space into a novel feature space, they end up losing interpretability and this transformation can be computationally expensive.

To formally introduce feature selection methods some notations must be defined. Given a dataset with n instances denoted by $\{\mathbf{x}_i\}_{i=1}^{n}$ from which we want to extract or select features. Each of these instances $\mathbf{x}_i$, is a d-dimensional vector (i.e. $\mathbf{x}_i \in \mathbb{R}^d$) which can be also represented by $\vec{\mathbf{x}}$. The instances can be represented by a matrix $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_n] = [\mathbf{x}^1, \cdots, \mathbf{x}^d]^T \in \mathbb{R}^{n \times d}$. The subscript ($\mathbf{x}_i$) and the superscript ($\mathbf{x}^j$) represent the i-th instance and the j-th feature, respectively [75].

Feature selection methods reduce the dimensionality of the data. Therefore, their goal is to map $\vec{\mathbf{x}} \in \mathbb{R}^d \to \vec{\mathbf{y}} \in \mathbb{R}^p$ where $p < d$. In feature selection, $\{\mathbf{y}^j\}_{j=1}^p \subseteq \{\mathbf{x}^j\}_{j=1}^d$, i.e. the selected subset of features in an inclusive subset of the original features [75].

2.5.1 Methods

Feature selection methods maps $\vec{\mathbf{x}} \in \mathbb{R}^d \to \vec{\mathbf{y}} \in \mathbb{R}^p$ where $p < d$. They work by removing features that are not relevant or redundant [36, 75, 76] without altering the original representation of the data [76]. With d dimensions the total number of possible subsets of features are 2^d. In consequence, if d is large it starts to be infeasible to compute all the exponential number of subsets in a reasonable time [75]. The evaluation of the subsets is based on some criterion, which can be separated into three categories:

1. the *filters* which select a subset of features from the dataset without any learning method;

2. the *wrappers* that uses a learning methods to evaluated which features will be selected for the subset

3. the *embedded techniques* which combine the feature selection and classification steps.

2.5.1.1 Filter methods

Filter methods perform the feature selection as a pre-processing step. It is independent from the learning stage and relies only on the attributes of the data [36]. Filter methods can be used in any machine learning algorithm and they are computationally inexpensive [16, 36, 76].. Despite the lower time consumption, one of the main disadvantages of filters is the fact that they do not interact with the learning method; which usually leads to worse performance when compared to other methods [16]. Nonetheless, they are able to eliminate irrelevant, duplicated, constant, redundant and correlated features [16, 36, 76].

They are divided into univariate and multivariate methods. Univariate considers each feature separately and selects the best features based on univariate statistical tests (ANOVA F-value, mutual information, $\mathcal{X}^2$). One problem that can occur with these methods is that they can select redundant variables, as they don't take the relationship between features into account. On the other hand, multivariate methods find relationship among features using the whole feature space. Therefore, these methods are able to handle duplicated, correlated and redundant features. [16, 36, 76].

Statistical and ranking filter methods are univariate, evaluating each feature individually. They evaluate whether each feature is important based on the distribution of the target labels. Essentially, these methods rank the features based on certain criteria methods and then select the features with the highest rank [75, 76].

Mutual information (MI) is a measure of the mutual dependencies between two variables. It measures the amount of information obtained about one variable by observing the other variable [75]. If X and Y are independent, their MI is zero. In feature selection, we need to find the MI between the feature and the target labels. In this case, since X and Y are dependent, the MI value is greater than zero and given by the relative entropy between the joint distribution and product distribution (Equation 2.1) [75].

$$MI(X;Y) = \sum_{x,y} P_{XY}(x,y) \log \left(\frac{P_{XY}(x,y)}{P_X(x) P_Y(y)} \right) \tag{2.1}$$

$\mathcal{X}^2$ test measures the dependence of feature occurrence on the target labels and is based on the $\mathcal{X}^2$ probability distribution [75]. It is commonly used for testing relationship between categorical variables (discrete finite values) with the variables usually being non-negative, typically Boolean, frequencies and counts [75]. The $\mathcal{X}^2$ measure for the j-th feature is obtained by Equation 2.2.

$$\mathcal{X}^2(\mathbf{x}^j, \mathbf{t}) = \sum_{p=1}^{p} \sum_{q=1}^{q} \frac{(n_{p,q} - \mathbb{E}_{p,q})^2}{\mathbb{E}_{p,q}} \tag{2.2}$$

where $n_{p,q}$ denotes the number of samples which have the p-th value of the j-th feature and the q-th value of the target label t; $\mathbb{E}_{p,q}$ is the expected value for $n_{p,q}$ and is obtained as:

$$\mathbb{E}_{p,q} = n \times Pr(p) \times Pr(q) = n \times \frac{\sum_{q=1}^{q} n_{p,q}}{n} \times \frac{\sum_{p=1}^{p} n_{p,q}}{n} \tag{2.3}$$

The largest $\mathcal{X}^2$ measures shows the most significant dependence between the feature and the target labels; therefore if it is below a pre-defined threshold the feature is discarded [75].

ANOVA Univariate test or ANalysis Of VAriance test measures the dependence between two variables. It assumes a linear relationship between the feature and the target labels and also that the feature is normally distributed. Therefore, it is well-suited for continuous features [77]. The largest ANOVA measures shows the most significant dependence between the feature and the target labels; therefore if it is below a pre-defined threshold the feature is discarded.

2.5.1.2 Wrapper methods

Wrapper methods use a learning algorithm as a subroutine, measuring the usefulness of each subset of features with the prediction performance of the learning algorithm over a validation set [36]. These are called greedy methods because they aim to find the best possible combination of features. Wrapper methods have the advantage of detecting the interaction between variables and they aim to find the optimal feature subset for the desired machine learning algorithm [75]. Although usually wrapper methods show a better performance when compared with filter methods, they have a much higher computation cost which increases as the number of features in the data increases [16]. The search methods can be divided into two categories: sequential selection and meta-heuristics.

Sequential selection methods creates the optimal feature subset by selecting the features from the given feature space in a sequential manner. There are four main categories: (i) Forward Feature Selection (FFS) which starts with no features and add one at a time; (ii) Backward Feature Selection (BFS) which starts with all features and removes one at a time; (iii) Exhaustive Feature Selection (EFS) which tries all the possible combinations; and (iv) Bidirectional Search (BS) does both forward and backward feature selection simultaneously. It is worth noticing that FFS and BFS can have a major drawback a part from the computational cost which is the impossibility to remove a feature after it was selected to be part of the subset [75].

Sequential wrapper backward elimination was introduced by Marill and Green [78] and implemented by Maciel-Guerra *et al.* (2019) [8] to evaluate how DS methods perform before and after the feature selection. The method initially considers the whole set of features. Subsequently, at each step, it drops a single feature whose absence causes the learning model to have the smallest mean validation error (root mean square error with cross-validation of 10 folds). For n available features, therefore, the first learning model will have n inputs. In the first iteration, n learning models are trained, where each model is comprised of a distinct set of $n - 1$ inputs. From one iteration to the next one, the size of the input vector is dropped by one. As it is difficult to detect when the minimum validation error is achieved and due to the fact that the error reduction does not occur monotonically, the elimination may proceed until a learning model with a single input is obtained. The remaining feature corresponds to that which is the most relevant. After analysing the curve [number of inputs] x [validation error], it is possible to choose the best subset of features producing the smallest mean validation error.

The backward elimination [78] is an approach that implements greedy decision steps, given that a large amount of subsets of features (candidate input vectors) are not tested. In backward elimination, once a feature is eliminated, no other subset containing this feature is assessed. This approach is locally optimal, at each elimination/insertion stage, although being clearly unable to predict all relevant complex interactions among features [79]. Despite the shortcomings of the method, their application guides to an acceptable compromise between cost and effectiveness. Furthermore, backward elimination appears more robust to local minima than constructive approaches, as it starts considering all the interactions among features [80], increasing the likelihood of retaining the relevant interactions at the final solution.

It is worth noticing that the WBS method admits any reasonable choice for the learning model, being an issue associated with the specific purpose of the application. Generally, linear learning models are adopted, given that a high number of learning models should be trained [80]. However, given that extreme learning machines (ELMs) [81] were designed to have a fast training phase, they are going to be adopted here. Moreover, since nonlinear models like ELMs are much more flexible than linear models, regularisation becomes a relevant issue here. Therefore, given our interest in the qualitative aspects of the obtained learning model, its interpretability and the fast training phase, we implement a regularised ELM as the learning model of WBS. To our knowledge, this is the first scalable approach in the literature that resorts to a non-linear embedded model.

Extreme learning machines (ELMs) [81] are multilayer neural networks with the hidden layer(s) projected in an unsupervised manner (being a random choice of weights a valid option). As emphasised by Kulaif and Von Zuben [82], the main advantages of ELMs are: (1) The training phase corresponds to solving a linear regression problem, which is far less computationally intensive when compared to the training process of MLPs, RBF neural networks, and SVMs; (2) There is no local minimum, given that the training is a convex optimisation problem, and even when a kernel must be defined, the influence of the kernel choice is reduced when compared to what is seen in SVMs; (3) Ridge regression founded on a single regularisation parameter is enough to promote the generalisation capability; (4) When properly defined according to general requirements, the number of neurons at the hidden layer and their weights, even when defined randomly, do not restrain the performance; (5) ELMs exhibits competitive generalisation performance when compared to MLPs, RBF neural networks and SVMs, being all of them universal approximators [83].

2.5.2 Challenges of small instance high-dimensional datasets

According to Zawbaa *et al.* (2018) [84] it is far more complex and challenging to deal with with high-dimensional small-instance datasets for two reasons: (i) the search space becomes too large; and (ii) the small number of instances do not provide enough examples to learn from. In another work, Kuncheva and Rodriguez (2018) [85] published an article on which they discussed a protocol for feature selection on small instance high-dimensional datasets. The authors raise two important questions: (i) "How reliable are any conclusions drawn for such datasets?" and (ii) "How meaningful is feature selection?".

Kuncheva and Rodriguez (2018) [85] showed that most papers published use a "contaminated" protocol, which includes the *peeking* phenomenon. The *peeking* happens if the test data is seeing during some part of the training step of the classifier or during the feature selection step. Most papers in the literature select a subset of features $\mathcal{S}$ by applying a feature selector $\mathcal{F}$ over all samples of the dataset. Next, the classifier model $\mathcal{C}$ is evaluated on the same data, most of the times using a cross-validation technique. The caveat here, according to the authors, is that the dataset is used twice: once for finding $\mathcal{S}$ through $\mathcal{F}$ and once for evaluating $\mathcal{C}$. Therefore, the testing data have already been used to select $\mathcal{S}$ which may cause a bias on the accuracy, i.e. the *peeking* phenomenon.

Kuncheva and Rodriguez (2018) [85] argue that the correct protocol should be to include the feature selector $\mathcal{F}$ into the cross-validation loop. A subset of features $\mathcal{S}_i$ is obtained for each cross validation fold; and the classifier $\mathcal{C}$ is trained on $\mathcal{S}_i$ using the same data as $\mathcal{F}$. The testing data is, then, used to evaluate $\mathcal{C}$ and the average accuracy over all the folds is estimated. At no point of this protocol the testing data is used neither by the classifier $\mathcal{C}$ or the feature selector $\mathcal{F}$.

The following section describes subspace clustering methods. This is methods can substitute the k-NN method to find the region of competences in the DS framework in high-dimensional datasets. Because they are able to select the most important features for a specific set of samples, which is a common phenomena in small instance high-dimensional datasets [10, 39, 86]. The motivation on why to use subspace clustering is further explained, as well some of the most important methods in the literature based on their contribution for the area.

2.6 Subspace Clustering

Clustering is an essential task for knowledge discovery [10, 39, 87]. Traditional clustering algorithms attempt to find clusters using similarity measures based on distance metrics [10, 39, 87]. Moreover, these methods use the whole set of features to compute the similarities [10, 39, 87]. However, with the advances in technology, data is increasingly growing in terms of its dimensions, which pose great challenges for clustering methods due to issues like distance concentration and hubness [40, 87]. Feature selection and extraction methods can be a good choice to decrease the dimensionality of the dataset. Nonetheless, according to Tian and Gu (2019) [87] some features might only work for a subset of samples and appear as noise for the rest of the samples, and this phenomenon is more common in high-dimensional datasets.

To deal with this problem, subspace clustering has been used by many authors to find clusters in different subsets of features and samples, as shown in the recent reviews by Parsons *et al.* (2004) [10], Kriegel *et al.* (2009) [39] and Muller *et al.* (2009) [86]. Clusters determined in subspaces can reduce the computational cost and provide a more relevant information regarding local structures of the feature space, which can assist establishing the most important features relevant to the end point investigated [10, 39, 87]. For microarray studies, for instance, the goal is to understand which biomarkers (features) are relevant to the biological activity.

2.6.1 Subspace clustering motivation

To illustrate the need for subspace clustering we describe here the analysis done by Parsons *et al.* (2004) [10] using a simple synthetic dataset (Figure 2.6). This dataset has four hundred samples spread out over four clusters with 100 samples each in 3 dimensions. The first two clusters (red and green) exist on dimensions a and b. They were created using a normal distribution with means 0.5 and -0.5 in dimension a and mean -0.5 in dimension b with standard deviation of 0.1. In dimension c, these two clusters have mean 0 and unit standard deviation. The other two clusters (blue and purple) exist in dimensions b and c. They were created using a normal distribution with means 0.5 and -0.5 in dimension c and mean 0.5 in dimension b with standard deviation of 0.1. In dimension a, these two clusters have mean 0 and unit standard deviation. If k-Means was applied in this data it would do a poor job of finding the clusters, this is because each cluster only exist in two dimensions and the third dimension is irrelevant. According to Parsons *et al.* (2004) [10], in higher dimension datasets this problems is even worse due to the number of irrelevant features and to issues such as distance concentration and hubness.

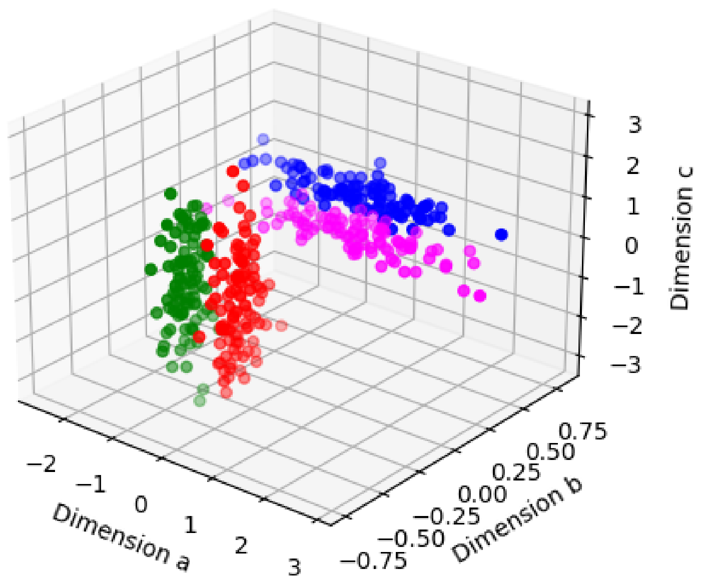


Figure 2.6: Example of a dataset with four clusters, each in two dimensions with the third dimension being noise [10].

For the synthetic dataset presented on Figure 2.6 if we try to use a feature selection technique will not work. Figure 2.7 shows the data projected into a single dimension and it is possible to observe that none of the three dimensions is sufficient to fully separate the four clusters. Alternatively, we could try to remove only one dimension. Figure 2.8 shows the 2D plot of the three possible combinations with a dataset with 3 dimensions.

In this again, none of the combinations yields in higher separation.

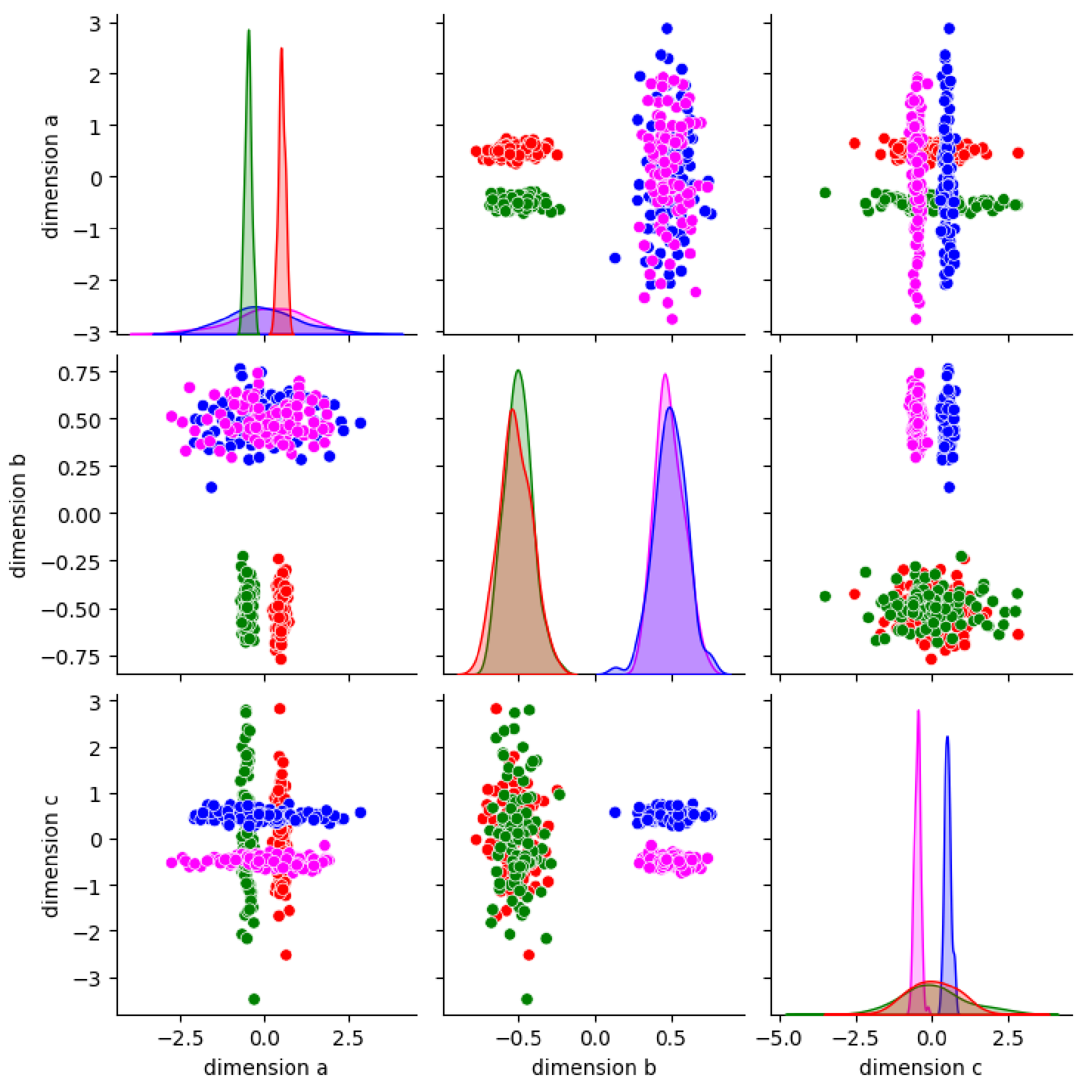


Figure 2.7: Histograms of each dimension of dataset presented on Figure 2.6 [10]

Nonetheless, it is worth noticing that the clusters red and green are easily separated from each other in dimensions a and b, while clusters blue and purple are easily separated using dimensions b and c. Therefore, the key of finding the clusters in the dataset is to use subspace clustering and find them in their appropriate subspaces.

2.6.2 Subspace clustering algorithms

This section discusses the main subspace clustering strategies and summarises some of the major subspaces clustering algorithms.

Parsons *et al.* (2004) [10] divided subspace clustering methods into two large groups:

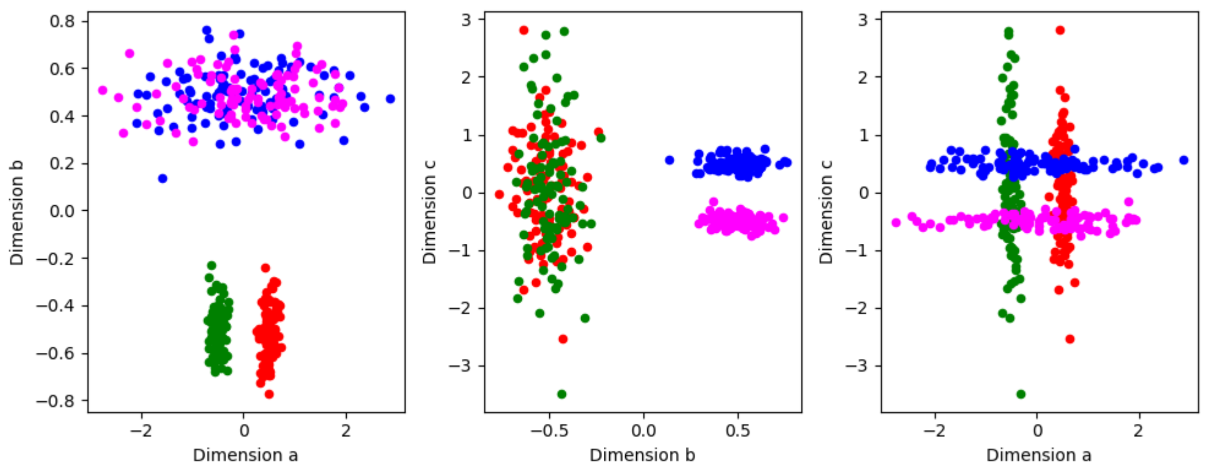


Figure 2.8: 2D plots of all possible combinations of dimensions of the dataset presented on Figure 2.6 [10]

bottom-up and top-down. A naive approach would be to search for all possible subspaces and use some cluster validation metric to determine the subspaces with the best clusters [10]. However, this problem is intractable and, therefore, more sophisticated heuristics are needed [10].

Bottom-up methods take advantage of the downward closure property of density to reduce the search space, i.e. if there are dense units in k dimensions, there are dense units in all $k-1$ dimensional projections. This algorithms first start by creating a histogram for each dimension and select the bins above a given threshold. Next, candidate two-dimensional subspaces are formed by combining only the dimensions which contain dense units forming clusters, dramatically reducing the search space. Higher dimensional subspaces are formed by combining adjacent dense units until there are no more dense units to be found. The nature of bottom up approaches can create overlapping subspaces where one instance is in zero or more subspaces. They are two main approaches to accomplish this: (i) CLIQUE and ENCUS use a static size grid to divide each dimension into bins; and (ii) MAFIA, Cell Based Clustering (CBF), CLTree and DOC use data driven strategies to determine the cut-points for the bins in each dimension (MAFIA and CBF use a histogram, CLTree uses a decision tree and DOC uses a maximum width and minimum number of instances per cluster) [10].

Top-down methods starts by finding clusters using all dimensions. Next, each dimension is assigned a weight for each cluster. These weights are then used to update the clusters. This approach requires expensive clustering algorithms in high-dimensional feature spaces. Top-down algorithms create clusters that are partitions of the dataset, therefore, each instance is assigned to only one cluster, so there are no overlaps in terms of instances. Parameter tuning is necessary to get meaningful results with the number of clusters and the size of the subspaces being the most critical ones. Most methods (PROCLUS, ORCLUS, FINDIT and δ-Clusters) determine the locality of the clusters by setting a weight to each dimension for each instance. COSA is a special algorithm that uses k-nearest neighbors for each instance to determine the weight of each dimension for the particular instance [10].

The SBDS framework proposed in the thesis use a bottom-up approach which has the advantage of reducing the search space, making these methods less computational expansive. Approaches of subspace clustering can also be divided into three major groups: cell-based, density based and clustering-oriented [86]. The SBDS framework was inspired on the work done by Tian and Gu (2019) [87]. The authors proposed a cell-based approach which divides the data into grid cells with a certain threshold and search clusters on the cells considering count of data points in these cells. Another important aspect of these methods is their pruning criterion for efficient subspace search based on a monotonicity property. This property states that given a subspace cluster in d-dimensions, all low dimension projections are also subspace cluster. Therefore, the negation of this property is used as a pruning criterion, i.e., if a set of objects does not form a subspace in a d-dimensional space then all higher dimensional projects do not form a subspace cluster either [86]. Although there are difference between the methods in this category, all of them share a main property of counting the number of objects inside a cell do determine if this cell is part or not of a subspace. This makes the algorithms more efficient, but can result in loss of information due to the discretization [86].

2.6.2.1 Subspace Clustering Based on Self-Organising Map (SCBSOM)

Recently, Tian and Gu (2019) [87] proposed a novel cell-based approach for subspace clustering based on self-organising maps called SCBSOM. This method aims to find cells using the trained SOM and allows overlapping between clusters. Clusters are first constructed for each dimension, and later merged together. SCBSOM search for clusters in the grid cells trained with SOM rather than the original sample space, as most of the traditional cell-based approaches. Moreover, since SOM preserves the topology of the data, it contributes to a higher accuracy when compared with other cell-based approaches. Tian and Gu (2019) [87] showed that SCBSOM had a better performance in 4 out of 7 datasets when compared with other subspace clustering methods and with SOM. The 7 datasets studied have always a higher number of samples when compared to the number of features (the number of features range from 5 to 75 and the number of samples range from 1500 to 5500).

The pipeline to select the subspaces presented by Tian and Gu (2019) [87] is implemented in the SBDS framework, with the main difference being that the authors used a SOM to find the subspaces whilst the SBDS propose the use of a Gaussian Kernel Density Estimator (GKDE) to get the one-dimensional clusters. The next section discusses how the subspace clusters can be selected.

2.7 Nearest Subspace Search

According to Basri *et al.* (2011) [88], the Nearest Subspace Search problem is defined as follows: let $\mathcal{S}^1, \mathcal{S}^2, \cdots, \mathcal{S}^n$ be a collection of subspaces in $\mathcal{R}^d$, each with an intrinsic dimension d_S retrieved from a dataset with m samples and d features. Given a query $\mathcal{Q}$ in $\mathcal{R}^d$, the distance between the query and the i^{th} subspace is $dist(\mathcal{Q}, \mathcal{S}^i)$. We seek the subspace $\mathcal{S}^*$ that is the nearest to the query, i.e., $\mathcal{S}^* = \arg\min_i dist(\mathcal{Q}, \mathcal{S}^i)$.

Computing the similarity between objects is an important task in data mining. Typically, a k-NN algorithm is used to find the nearest samples of a query in a dataset using a distance function. However, they consider all dimensions when computing the distance between data points. For datasets with a high number of dimensions, similarity measures, including Euclidean distance, loose their discriminative ability [17, 89].

Basri *et al.* (2011) [88] proposed a mathematical approach to calculate an approximate distance between points and subspaces from subset dimensions to tackle high-dimensional data. Their experiments indicate that an approximate nearest subspace can be located faster than the exact nearest subspace, with little loss of accuracy, in large databases.

In 2015, Hund *et al.* [89] introduced the concept of Subspace Nearest Neighbour Search (SNNS). It aims at finding the nearest neighbours of a sample in a relevant subspace. The paper proposes three questions: (1) ”What is a relevant subspace for a given query?”; (2) ”How can we computationally extract this relevant information?”; and (3) ”How can we adapt ideas from subspace clustering, outlier detection, or feature selection for SNNS?”. These three questions were used to develop the core framework of this thesis. The SBDS framework proposes a subspace clustering approach to identify the most relevant subspaces in the data making them the Regions of Competence of the framework; and uses a SNNS approach to select the most important subspaces for each unseen test sample.

2.8 Chapter Summary

This chapter started with multiple classifier system to introduce the basic concepts for dynamic selection of classifiers. We indicated the reasons on why this is an important area to study and its advantages showcased by the literature. Subsequently, we identified the challenges of working with high-dimensional datasets and how the literature overcomes them by using dimensionality reduction. Next, the chapter reviewed subspace clustering and nearest subspace search methods showing their motivations and advantages to be used on high-dimensional datasets.

2.9 Conclusions from the literature

From the literature, we identified the need to improve the overall DS framework to address high-dimensional data. DS methods are well known in the literature for their superior performance when compared with traditional classification methods and static selection. However, to the best of our knowledge, there is no work evaluating these methods for high-dimensional small instance datasets.

Although traditional feature selection methods are able to remove features that are irrelevant and/or redundant, they fail to address the issue properly because in high-dimensional data a phenomena called local feature relevance happens, ie.e different subsets of features are relevant for different samples. Hence, the choice of incorporating a subspace clustering method into the SBDS framework, because this approach has been shown to be extremely effective in detecting the most important aspects of the data not only in terms of features but also in terms of the samples.

The next chapter (Chapter 3) evaluate high-dimensional, small instance datasets in different scenarios comparing traditional classification methods with dynamic selection methods.

Chapter 3

Dynamic selection applied to high-dimensional datasets

***

3.1 Introduction

In data classification, *observations* are instances of a particular phenomena (e.g. clinical patient measurements), each one being a vector of values measured on *variables* (e.g. blood pressure, height, weight, heart rate) [13]. In recent years, technology enabled researchers to gather increasingly amounts of data not only in terms of observations, but also in the number of variables [17, 41]. Financial, risk management, computational biology, health studies are areas where high-dimensional datasets are produced. However, in some of these areas, such as biology and medicine, it might not be feasible to have thousands or millions of samples due to the nature of the disease or the access to samples [17, 41].

Yang *et al.* (2015) [90] identified the following five major challenges when working with high-dimensional data:

1. Distance concentration: denotes the tendency of distances between all pairs of points to become almost equal

2. Due to the high number of features compared to the small number of samples, these datasets tends to have higher sparcity. In addition, it is likely that correlations exist between the different dimensions, and, therefore, the most important features are difficult to define.

3. Datasets tends to be unstructured. In addition, noise and uncertainties often exist, which impose challenges to its pre-processing and applying data mining techniques. Classification algorithms tend to be problem-specific and in some cases even data-specific.

4. As the number of features increases, the possible combinations of clusters increases exponentially and clustering can become a NP-hard problem.

5. Even with the recent increase in speed of modern computers and cheaper parallel and cloud computing, the increasing number of features are still a problem and efforts on developing new classification methods are needed.

Therefore, this chapter first introduces in more details the issues regarding hubness and distance concentration in high-dimensional datasets and evaluate small instance high-dimensional datasets to understand if they suffer from these issues. Next, the chapter investigates the challenges classifiers face when dealing with high-dimensional datasets by creating synthetically data varying the number of features. The results indicate that when the number of features increase, single classifiers and DS methods have a drop in their performance. Finally, DS methods are compared with majority voting and singles classifiers before and after two feature selection approaches: filter and wrapper.

3.2 Hubness and distance concentration analysis

Datasets with a high number of features usually poses challenges that are commonly referred as the “curse of dimensionality”. One of the main issues is distance concentration which is the tendency of distance to all points to become almost equal in high-dimensional spaces [20, 25, 40]. This directly affects machine learning application, specially those approaches based on distance metrics, such as k-NN. There is another important aspect of the “curse of dimensionality” which is related to nearest neighbours. Let $D \subset \mathbb{R}^d$ be a set of d-dimensional points and $N_k(\vec{\mathbf{x}})$ the number of *k-occurrences* of each point $\vec{\mathbf{x}} \in D$, i.e., the number of times a point $\vec{\mathbf{x}}$ appears among the k-nearest neighbours of all points in D, according to some distance metric [20]. According to Radovanovic *et al.* (2010) [20], as d increases, the distribution of N_k becomes considerably skewed to the right, resulting in the appearance of *hubs*, i.e., points that are “popular” nearest neighbours.

Radovanovic [20, 40] showed that hubness is an inherent property of high-dimensional datasets with negative influence on classification algorithms, such as k-NN, Suport Vector Machines with RBF kernels and Adaboost [20, 40]. In addition, the authors showed that hubness has a negative influence on clustering methods, because intra-clusters distance may be increased due to points with low k-occurrences, since they act as outliers [20, 40, 91]; also hubs do not cluster well because they are close to many points, including other clusters [20, 40, 91].

The above findings in relation to hubness and distance concentration are relevant to machine learning because many algorithms use distance between data points. First, a an illustrative experiment is conducted to demonstrate the changes in the distribution of N_k with varying dimensionality. By using a random synthetic dataset with 10000 samples and d features independently drawn from a normal distribution and the following distances: Euclidean, Squared Euclidean and cosine. Figure 3.1 shows empirically the distributions of N_7 for (a) $d = 3$, (b) $d = 20$ and (c) $d = 100$. As the dimensionality increases, the

N_7 distributions become skewed to the right, i.e., the majority of the points are on the tails of the distribution. The skeweness, according to Radovanovic *et al.* (2010) [20] is an indication of the hubness phenomenon, that is, points with high k-occurrences.

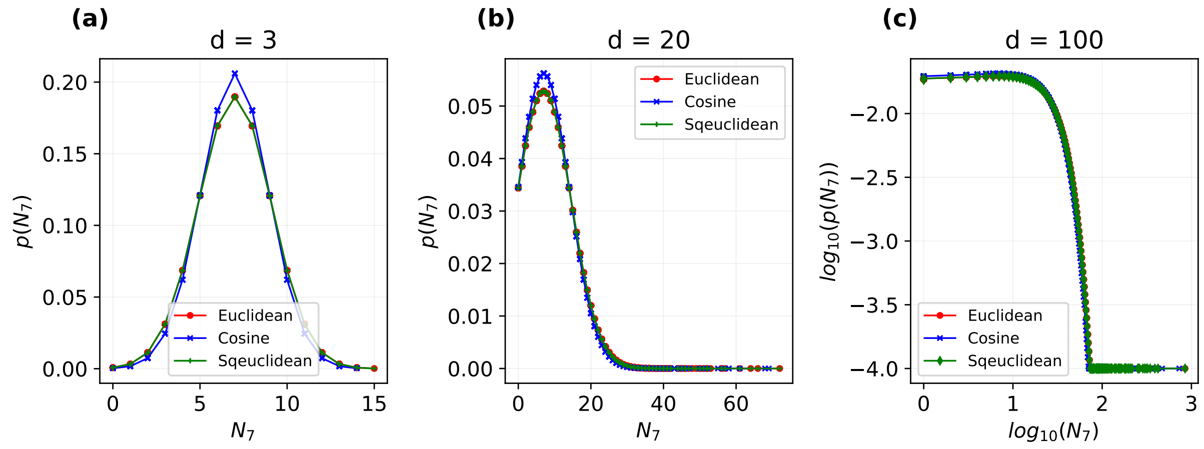


Figure 3.1: Empirical distribution of N_7 for Euclidean, Squared Euclidean and cosine distances for normally distributed datasets with $n = 10000$ and dimensionality (a) $d = 3$, (b) $d = 20$ and (c) $d = 100$

To illustrate the hubness phenomenon on real data, 11 real-world benchmark datasets with a high number of features and a low number of samples were examined to verify if they suffer from hubness issues. Four datasets are text based while seven are biomedical datasets relating to different types of cancer (lung, prostate, leukemia, colon, glioma and ovarian). The attributes of each dataset are shown in Table 3.1. This study attempts to explain the phenomena of hubness and distance concentration and the mechanisms through which hubs emerge.

Table 3.1: datasets attributes

dataset	**sample (n)**	**features (d)**	**ratio (n/d)**	**No. of classes**	**Distribution**	**Type**
ALLAML [92]	72	7129	0.0101	2	65.3 - 34.7%	Microarray
Arcene [92]	200	10000	0.02	2	56 - 44%	Mass Spectrometry
Basehock [92]	1993	4862	0.4099	2	49.9 - 50.1%	Text
Colon [92]	62	2000	0.031	2	64.5 - 35.5%	Microarray
Dexter [93]	600	20000	0.03	2	50 - 50%	Text
Gli85 [92]	85	22283	0.0038	2	30.6 - 69.4%	Microarray
Leukemia [92]	72	7070	0.0102	2	65.3 - 34.7%	Microarray
Pcmac [92]	1943	3289	0.5907	2	50.5 - 49.5%	Text
Prostate [92]	102	5966	0.0171	2	49 - 51%	Microarray
Relathe [92]	1427	4322	0.3302	2	54.6 - 45.4%	Text
Smk-Can [92]	187	19993	0.0094	2	48.1 - 51.9%	Microarray

Table 3.2 lists the main findings in terms of statistics which can describe if hubness

appears in this datasets for three different distance metrics (Euclidean, Cosine and Canberra). The number of nearest neighbours k was fixed to 7, as this is the number most papers on DS methods use to determine the region of competence. The columns on this table describe the following dataset characteristics:

Table 3.2: 10 real-world datasets characteristics for three different distance metrics (Euclidean, Squared Euclidean and Cosine)

dataset	**Distance**	**n**	**d**	d_{mle}	S_{N_k}	p-**value** S_{N_k}	**7-NN Acc**	**Clu**	$C_{cm}^{N_7}$	p-**value** $C_{cm}^{N_7}$	$C_{dm}^{N_7}$	p-**value** $C_{dm}^{N_7}$
Allaml	Euclidean	72	7129	20,65284	1,285764	7,6E-05	77,86%	2	-0,85501	1,21E-21	-0,81643	2,34E-18
Allaml	Cosine	72	7129	10,83564	1,185723	0,000192	80,54%	2	-0,87825	4,03E-24	-0,8092	7,93E-18
Allaml	Sqeuclidean	72	7129	10,32642	1,285764	7,6E-05	77,86%	2	-0,85501	1,21E-21	-0,81643	2,34E-18
Arcene	Euclidean	200	10000	19,62199	0,778613	2,95E-05	85,00%	12	-0,60045	5,57E-21	-0,37683	3,8E-08
Arcene	Cosine	200	10000	10,02271	1,016812	2,43E-07	82,00%	11	-0,69375	4,87E-30	-0,30282	1,31E-05
Arcene	Sqeuclidean	200	10000	9,810997	0,778613	2,95E-05	85,00%	12	-0,60045	5,57E-21	-0,37683	3,8E-08
Basehock	Euclidean	1993	4862	50,24644	19,02935	0	60,96%	44	-0,4162	2,47E-84	-0,32971	9,52E-52
Basehock	Cosine	1993	4862	10,2059	1,373365	3,05E-83	91,62%	44	-0,1409	2,65E-10	0,024623	0,271894
Basehock	Sqeuclidean	1993	4862	25,12322	19,02935	0	60,96%	44	-0,4162	2,47E-84	-0,32971	9,52E-52
Colon	Euclidean	62	2000	11,79957	0,179461	0,529365	79,52%	3	-0,80951	1,67E-15	0,135635	0,293203
Colon	Cosine	62	2000	7,532528	1,978379	6,39E-07	74,29%	7	-0,78223	6,06E-14	-0,77359	1,7E-13
Colon	Sqeuclidean	62	2000	5,899783	0,179461	0,529365	79,52%	3	-0,80951	1,67E-15	0,135635	0,293203
Dexter	Euclidean	600	20000	55,62183	9,269779	1E-150	51,00%	13	-0,43	2,13E-28	-0,41378	3,22E-26
Dexter	Cosine	600	20000	87,22038	3,182544	7,62E-67	88,33%	2	-0,71551	3,21E-95	-0,71915	1,29E-96
Dexter	Sqeuclidean	600	20000	27,81091	9,269779	1E-150	51,00%	13	-0,43	2,13E-28	-0,41378	3,22E-26
Gli	Euclidean	85	22283	21,20165	1,655786	4,98E-07	80,83%	3	-0,90784	4,49E-33	-0,86622	9,81E-27
Gli	Cosine	85	22283	11,00125	1,082364	0,000195	82,08%	3	-0,86363	2,06E-26	-0,76956	7,66E-18
Gli	Sqeuclidean	85	22283	10,60083	1,655786	4,98E-07	80,83%	3	-0,90784	4,49E-33	-0,86622	9,81E-27
Leukemia	Euclidean	72	7070	25,97378	0,811738	0,005734	85,89%	5	-0,75997	9,88E-15	-0,42996	0,000164
Leukemia	Cosine	72	7070	12,74763	1,285482	7,62E-05	92,86%	3	-0,91665	1,37E-29	-0,69849	9,02E-12
Leukemia	Sqeuclidean	72	7070	12,98689	0,811738	0,005734	85,89%	5	-0,75997	9,88E-15	-0,42996	0,000164
Pcmac	Euclidean	1943	3289	50,79432	17,37004	0	68,40%	43	-0,31288	2,18E-45	-0,22186	4,3E-23
Pcmac	Cosine	1943	3289	12,73431	14,31757	0	77,25%	41	-0,25198	1,6E-29	-0,1024	6,11E-06
Pcmac	Sqeuclidean	1943	3289	25,39716	17,37004	0	68,40%	43	-0,31288	2,18E-45	-0,22186	4,3E-23
Prostate	Euclidean	102	5966	13,62113	0,391625	0,095436	80,45%	8	-0,52107	1,97E-08	-0,29431	0,002677
Prostate	Cosine	102	5966	6,908708	0,279018	0,228342	80,36%	3	-0,74048	5,88E-19	-0,17279	0,082436
Prostate	Sqeuclidean	102	5966	6,810564	0,391625	0,095436	80,45%	8	-0,52107	1,97E-08	-0,29431	0,002677
Relathe	Euclidean	1427	4322	41,44929	15,91792	0	72,10%	25	-0,47938	7,04E-83	-0,42594	5,59E-64
Relathe	Cosine	1427	4322	10,94281	3,716883	2,5E-175	86,47%	32	-0,15917	1,48E-09	-0,00542	0,83793
Relathe	Sqeuclidean	1427	4322	20,72464	15,91792	0	72,10%	25	-0,47938	7,02E-83	-0,42594	5,59E-64
Smkcan	Euclidean	187	19993	17,09152	1,202981	1,42E-08	63,48%	10	-0,77918	2,2E-39	-0,53401	3,54E-15
Smkcan	Cosine	187	19993	8,59914	1,109409	9E-08	61,32%	10	-0,78852	6,5E-41	-0,48767	1,45E-12
Smkcan	Sqeuclidean	187	19993	8,545761	1,202981	1,42E-08	63,48%	10	-0,77918	2,2E-39	-0,53401	3,54E-15

[1] Number of samples n and features d
[2] d_{mle}: estimated intrinsic dimensionality
[3] S_{N_k}: the asymmetry of N_k as characterised using the standardised third moment $S_{N_k} = E(N_k - \mu_{N_k})^3/\sigma^3_{K_k}$, where μ_{N_k} and σ_{N_k} are the mean and the standard deviation of N_k respectively.
[4] p-value S_{N_k}: is the corresponding p-value of S_{N_k}, which tests whether the skewed data is different from the normal distribution.
[5] 7-NN Acc: 7-Nearest Neighbour mean accuracy using a 10-fold cross validation.
[6] Clu: number of clusters determined with K-Means clustering by exhaustive search of values between 2 and $\sqrt{n}$, where n is the number of samples, to maximise $C_{cm}^{N_7}$
[7] $C_{cm}^{N_7}$: is the Spearman correlation of the observed N_k and the distance to the closest group mean.
[8] p-value $C_{cm}^{N_7}$: the two-sided p-value for a hypothesis test whose null hypothesis is that two sets of data (N_7 and $C_{cm}^{N_7}$) are uncorrelated.
[9] $C_{dm}^{N_7}$: is the Spearman correlation of the observed N_k and the distance to dataset mean (centre).
[10] p-value $C_{dm}^{N_7}$: the two-sided p-value for a hypothesis test whose null hypothesis is that two sets of data (N_7 and $C_{dm}^{N_7}$) are uncorrelated.

It can be observed that for 9 datasets there is a 99% confidence level that the S_{N_k} distribution is different from a normal distribution. Only the Prostate and the Colon datasets have a S_{N_k} with a p-value higher than 0.01. Also some datasets present very high S_{N_k} values, indicating strong hubness on the corresponding datasets. Moreover, in most cases $C_{cm}^{N_7}$ is much stronger than $C_{dm}^{N_7}$, i.e., *hubs* are much closer than other points to their respective cluster centres. It is worth noticing as well the estimated intrinsic dimensionality is much lower than the real dimensionality of the datasets.

In order to understand the mechanisms behind hubs formation its necessary to understand (1) the geometrical and distribution setting in which some points tends to be closer than the dataset mean and (2) why such points become hubs. To illustrate this, the distribution of the Euclidean distance of all points to the true data mean is presented on panels (c) and (d) of Figures 3.2 and 3.3. According to Radovanovic *et al.* (2010) [20], it is known and expected for points that are closer to the mean of the data to also be closer, on average, to all other points. Panels (c) and (d) of Figures 3.2 and 3.3 and Appendix C indicate that this tendency is amplified in high-dimensional datasets, with points that reside in the proximity of the data mean become closer to all other points. This results indicates how the hubness and the distance concentration phenomena are related.

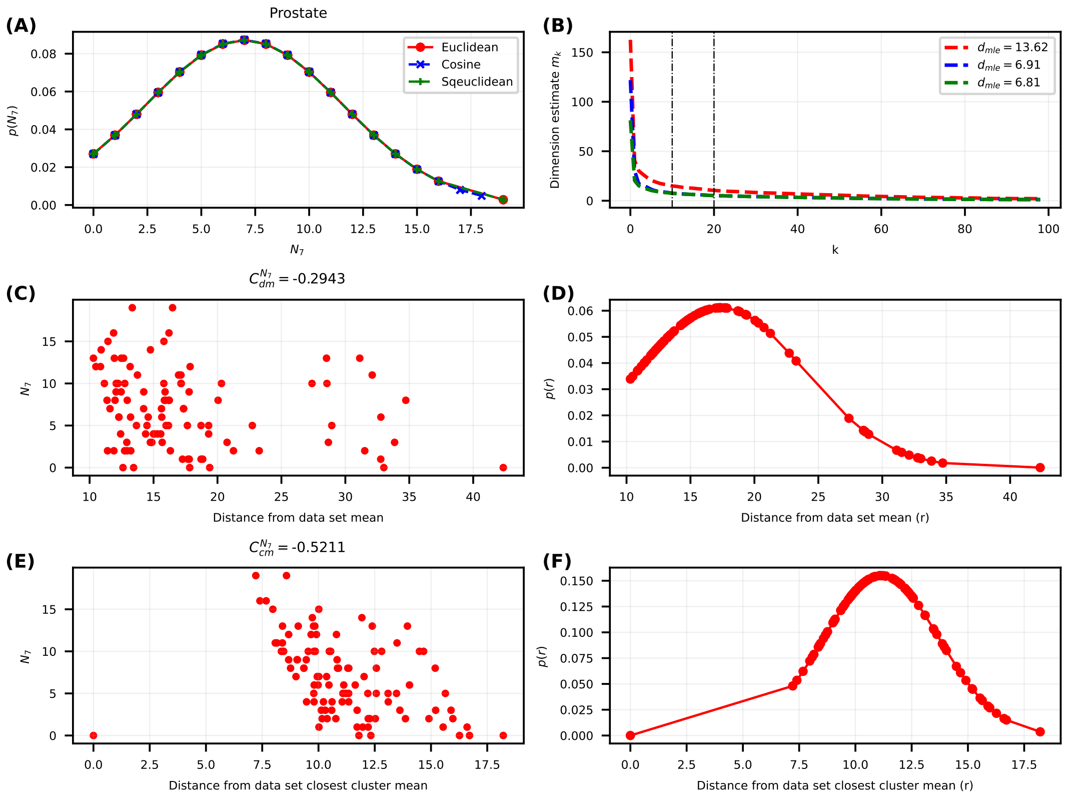


Figure 3.2: Hubness analysis of Prostate dataset. Six indicators have been used to evaluate the hubness phenomenon: (A) empirical distribution of N_7 for Euclidean, square Euclidean and cosine distances; (B) intrinsic dimensionality analysis; (C) scatter plot and spearman correlation of N_7 against the Euclidean distance to the dataset mean; (D) probability density function of observing point at distance r from the mean of the dataset; (E) scatter plot and spearman correlation of N_7 against the Euclidean distance to the closets cluster mean; (F) probability density function of observing point at distance r from the closets cluster mean.

To examine the clustering attributes of real world datasets, Table 3.2 shows: (i) the spearman correlation, denoted as $C_{dm}^{N_7}$ of the observed N_7 and the distance from the dataset mean, and (ii) the spearman correlation, denoted as $C_{cm}^{N_7}$, of the observed N_7 and the distance to the closest group mean. Groups are determined with K-means clustering where the number of clusters is determined by exhaustive search between 2 and $\sqrt{n}$, to maximise $C_{cm}^{N_7}$, with n being the number of samples. In most cases, $C_{cm}^{N_7}$ is stronger (closer to -1) when compared to $C_{dm}^{N_7}$, which indicates that hubs tends to be closer that other points to their respective cluster centres. Panels (e) and (f) of Figures 3.2 and 3.3 and Appendix C, shows the scatter plot and the empirical distribution of distances from the closest cluster mean. In all the cases, the probability of observing a point near the centre becomes closer to zero.

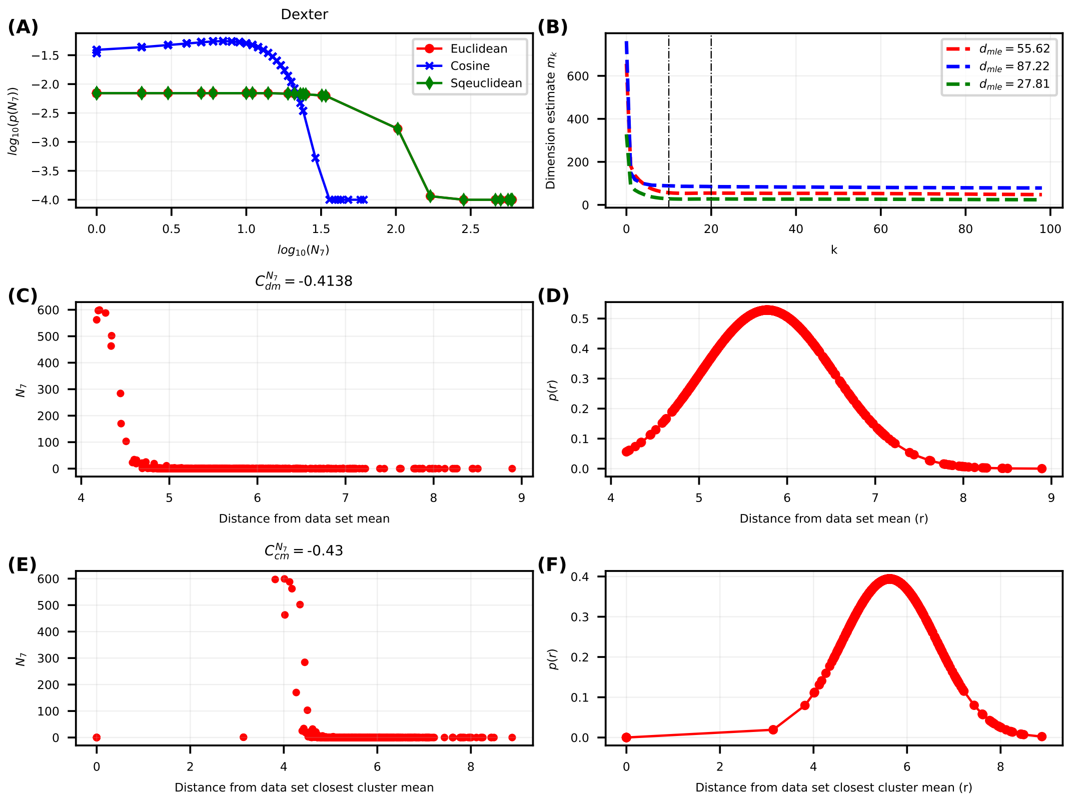


Figure 3.3: Hubness analysis of Dexter dataset. Six indicators have been used to evaluate the hubness phenomenon: (A) empirical distribution of N_7 for Euclidean, square Euclidean and cosine distances; (B) intrinsic dimensionality analysis; (C) scatter plot and spearman correlation of N_7 against the Euclidean distance to the dataset mean; (D) probability density function of observing point at distance r from the mean of the dataset; (E) scatter plot and spearman correlation of N_7 against the Euclidean distance to the closets cluster mean; (F) probability density function of observing point at distance r from the closets cluster mean.

3.3 DS applied to high-dimensional datasets: pool generation and comparison with MCS methods

In the majority of DS publications, the pool of classifiers is generated using the bagging method, which is a well known generation approach for static methods [4, 29, 67]. In addition, some DS methods used a pool of classifiers made by heterogeneous methods, which were compared with a homogeneous pool [29, 67]. The authors indicated that, although the expected diversity in the heterogeneous pool could increase the performance, in reality, the homogeneous outperformed the heterogeneous one, because at the majority of the times the same classifiers were selected in the heterogeneous pool and the methods chosen were not complementary.

Diversity, therefore, is still an open question in the DS area and it should be more carefully studied. The SBDS and cSBDS methods proposed on Chapter 5 generate intrinsically diverse classifiers, since each classifier is trained only in one specific region of the feature space, determined by the subspace clustering method. Nonetheless, future work on increasing the diversity by using different classifiers in different regions could perhaps help to improve the performance of these two methods.

To compare the different DS methods on high-dimensional datasets a bagging approach is used due to its simplicity and because most of the studies in the area of DS have focused on finding the neighbourhood of a new data instance and choosing the most competent classifier(s) to make the prediction. According to Yasar *et al.* [73] the focus on having a diverse ensemble of classifiers was ignored on previous studies because the DS systems is not required to generalise to all samples, i.e. since the system focus on prediction the class of a single instance at each time, the classifiers' performance over the different regions is not relevant.

Therefore, a pool of classifiers composed of 11 decision trees is used, as suggested by Woloszynski *et al.* [67], with pruning level set to 10. The pool is generated using the bagging technique, similarly to the methodology followed by Woloszynski in [29, 67]. An odd number of classifiers is chosen to overcome decision ties. These classifiers are used due to their instability when trained with different sets of data, i.e., small differences on the training set can create different trees [67]. Moreover, a study was also conducted to compare the pool of classifiers composed with decision trees, with a pool of 11 perceptrons and an pool of 11 naive bayes. Using the "gentrunk" function from Matlab PRTools with number of samples fixed at 300 and number of features ranging from 2 to 5000, we have created 224 datasets and evaluated 21 DS methods. The DS methods were ranked according to their accuracy. The pool of classifiers composed with 11 decision trees had an overall rank of 1.73, while the pool with 11 nayve bayes had an overall rank of 2.16 and the perceptrons an overall rank of 2.75.

In addition, the performance of an Adaboost (Figure 3.4) and a Random Forest (Figure 3.5), state-of-art multiple classifier systems, were tested on the datasets presented on Table 3.1. The accuracy, sensitivity and specificity are shown in the violin plots which is similar to a box plot, but also indicates the probability density of the data at different values, smoothed by a kernel density estimator. It is interesting to observe that in both analysis three of the four datasets with the highest number of dimensions had the lowest accuracy performances.

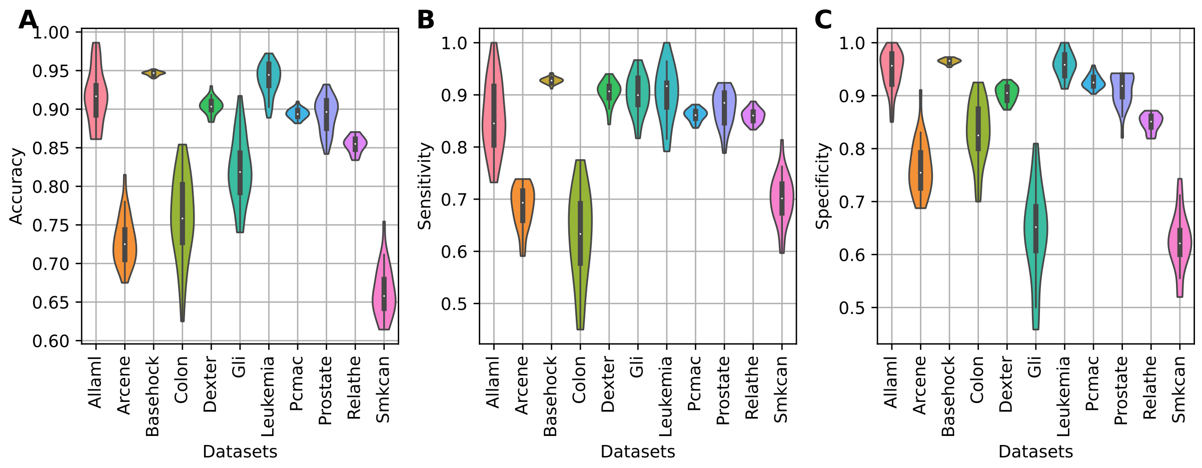


Figure 3.4: Supervised machine learning prediction of 11 high-dimensional small instance datasets. Prediction performance results of Adaboost classifier. Three performance indicators have been used to evaluate the classification: (A) accuracy, (C) sensitivity and (D) specificity value from 30 runs.

The next three section moves to discuss experiments that showcase the issues of using DS methods in high-dimensional small instance datasets. We first apply the DS

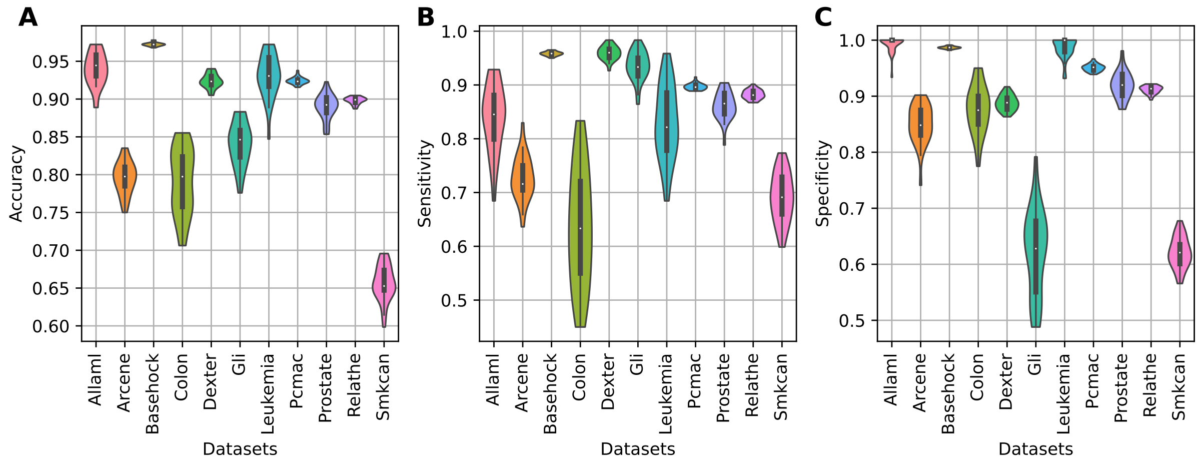


Figure 3.5: Supervised machine learning prediction of 11 high-dimensional small instance datasets. Prediction performance results of Random Forest classifier. Three performance indicators have been used to evaluate the classification: (A)accuracy, (C) sensitivity and (D) specificity value from 30 runs.

methods on synthetic data. Since the results demonstrated that when increasing the number of features the performance of dynamic selection is reduced, we decided to apply DS methods on real-world small instance high-dimensional datasets and compared them to majority voting and single classifiers. The results presented here were also described on Maciel-Guerra *et al.* (2019) [8] and on Maciel-Guerra *et al.* (2020) [9].

3.4 DS on synthetic data

In this section, DS is evaluated using several synthetic datasets with different number of samples and number of features were generated based on the Trunk's classification problem [94]. These datasets are based on a two-class problem of normally distributed data. The features are independent and have a unit standard deviation for both classes. The class averages are given by $(\frac{1}{i})^{\frac{1}{2}}$ and $-(\frac{1}{i})^{\frac{1}{2}}$ for the i-th feature [94]. Figure 3.6 shows the first 7 features for a problem with 3000 samples.

3.4.1 Experimental Methodology

To analyse if dynamic selection methods and specially the methods to define the region of competence (k-Nearest Neighbour and k-Means) will suffer from the curse of dimensionality in high-dimensional datasets and have their performance decreased, we

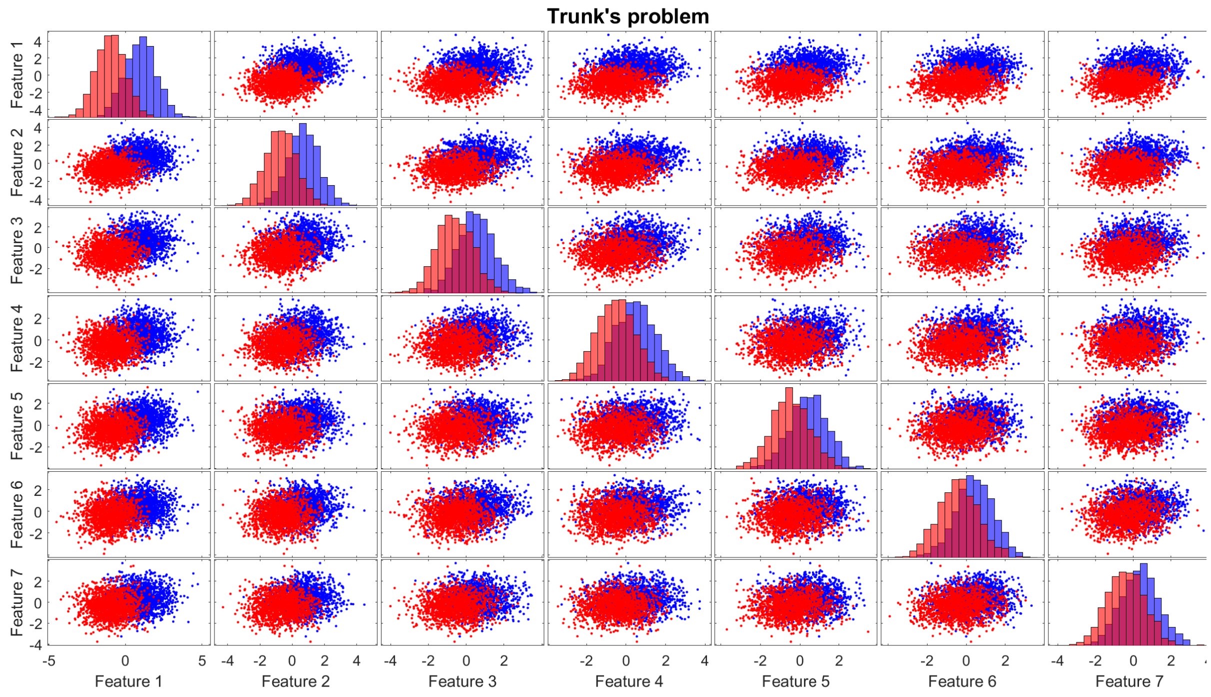


Figure 3.6: Trunk's classification problem of normally distributed data. The features are independent and all have a unit variance for both classes. First 7 features are shown for a problem with 3000 samples.

decided to created multiple Trunk's datasets, with the number of samples ranging from 200 to 1000 and features ranging from 2 to 5000. The pool of classifiers were created using the bagging algorithm with a decision tree as base classifier. The size of the pool of classifiers was 11. The experiments were run 10 times using 50% of the samples as the training set, 25% as validation set and 25% as testing set. The size of the region of competence was set to 7 as proposed by Cruz *et al.* (2017) [4].

3.4.2 Results

Figures 3.7 and 3.8 show the mean accuracy of a Decision Tree classifier and a k-NN, respectively, computed using a 10-fold cross validation for datasets generated with the "gentrunk" function from Matlab PRTools with number of samples raging from 200 to 1000 and number of features ranging from 2 to 5000. From both figures, we observe that for a low number of features and a high number of samples the accuracy is high, and for a high number of features and a low number of samples the accuracy is significantly reduced, which indicates that the both the Decision Tree and the k-NN classifiers suffered from the "curse of dimensionality" in this context.

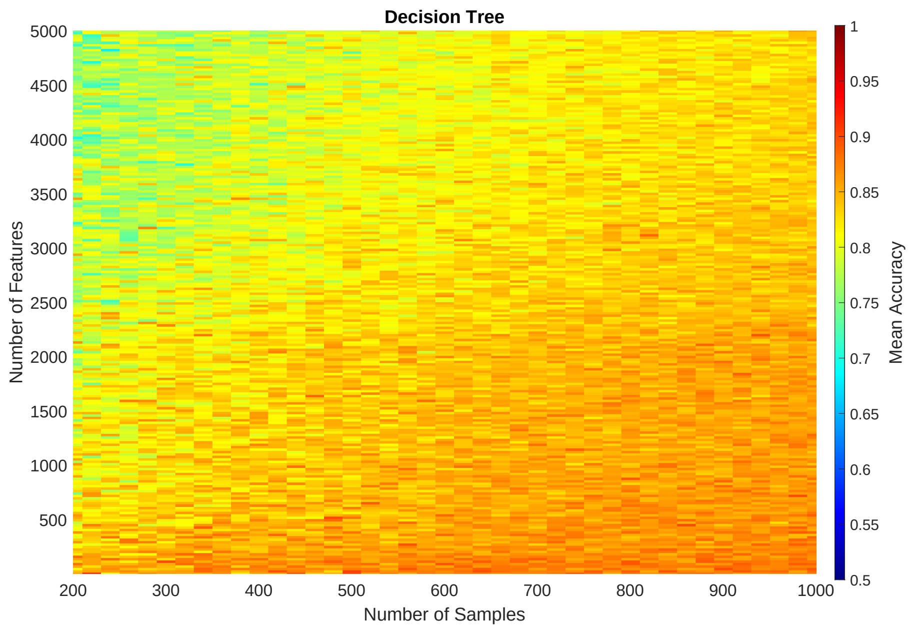


Figure 3.7: Mean accuracy of Decision Tree computed using a 10-fold cross validation for datasets generated with the "gentrunk" function from Matlab PRTools with number of samples raging from 200 to 1000 and number of features ranging from 2 to 5000

Since the single classifiers have their performance deteriorated when the numbers of features increase, we hypothesise that the same thing could happen with DS methods. Therefore, to verify this hypothesis we tested the DS methods indicated on Table 2.1 on 224 datasets with features ranging from 2 to 5000 (99 datasets with the number of features ranging from 2 to 100, 45 with the number of features raging from 120 to 1000 and 80 datasets with the number of features ranging from 1050 to 5000) and the number of samples fixed at 300. The results are shown in Figure 3.9, where it is possible to observe that the methods that rely on the k-Nearest Neighbour and k-Means to define the region of competence had their performance decreased with the increase on the number of features. DES-EXP, DES-KL and DES-MD, that use the entire validation set to compute the confidence level of each base classifier, also had their performances decreased but with a lower rate, which may be due to the use of Euclidean distance to calculate the potential function. META-DES and DSOC also had their performances decreased in a lower rate,

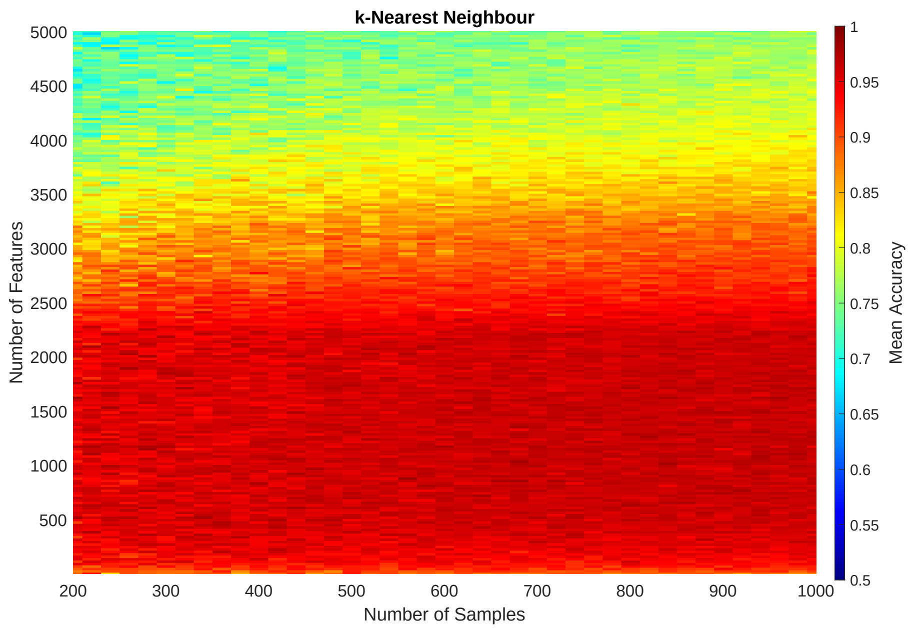


Figure 3.8: Mean accuracy of k-NN computed using a 10-fold cross validation for datasets generated with the "gentrunk" function from Matlab PRTools with number of samples raging from 200 to 1000 and number of features ranging from 2 to 5000

even though they use a k-Nearest Neighbour to define the region of competence; these results can be explained by the fact that they combine different types of features to measure the confidence level; therefore, even if one of these features fail, the others can carry the information necessary to predict the correct label.

3.4.3 Conclusions

To evaluate how DS methods perform on high-dimensional synthetic datasets, first datasets with increasing number of features and different number of samples were created using the "gentrunk" function from the PRTools toolbox. First, the performance of a decision tree and a k-NN were evaluated and we observed that their performance is decreased when the number of features increase, which confirms the issues with the "curse of dimensionality". Afterwards, we evaluated 224 datasets with 300 samples and features ranging from 2 to 5000. We observed that all DS methods had a decrease in performance, with methods that had a k-NN or a k-Means to define the region of competence being the ones with the highest decrease in performance. On the other hand, DES-KL, DES-EXP

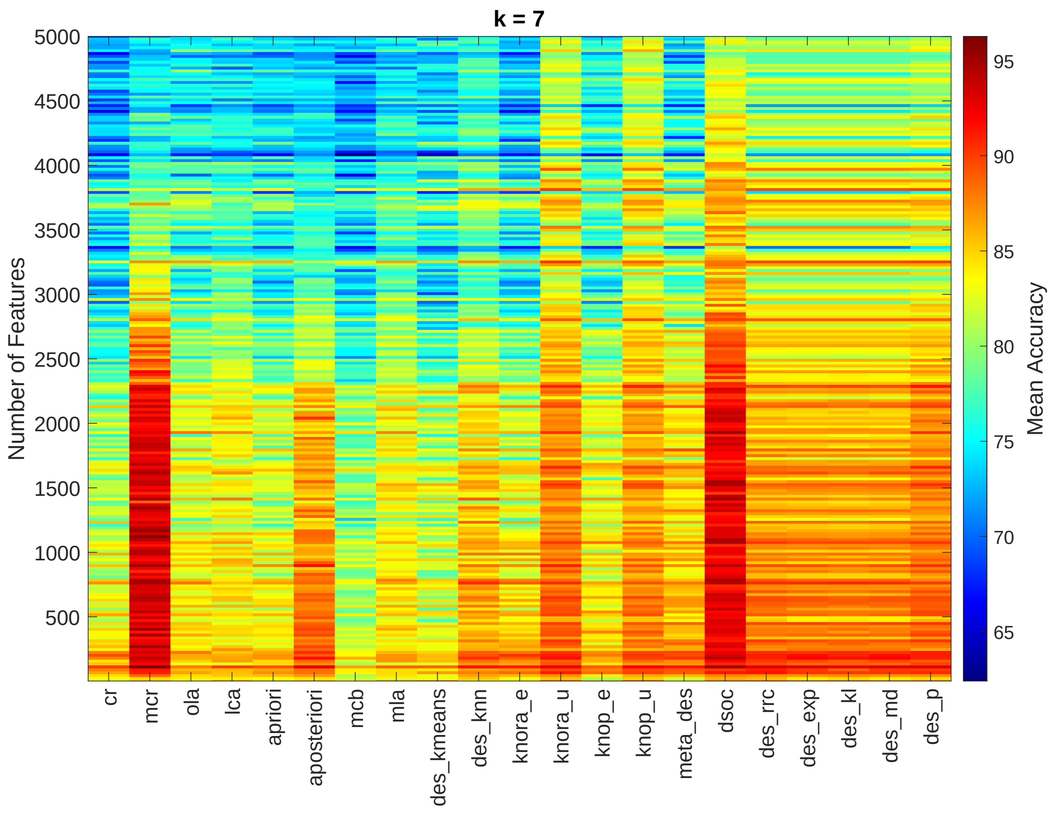


Figure 3.9:]
Mean accuracy of 21 DS methods computed using a 10-fold cross validation for datasets generated with the "gentrunk" function from Matlab PRTools with 300 samples and number of features ranging from 2 to 5000. The size of the region of competence was set to 7 and the pool of classifiers was composed of 11 decision trees.

and DES-MD had a smaller decrease in performance because they use the entire validation set to compute the confidence level of each base classifier. Hence, we can conclude that DS methods also suffer from the "curse of dimensionality" when used on datasets with a high number of features and a low number of samples. The next section investigates how the DS methods perform on small instance high-dimensional datasets from real world problems and compares them to the majority voting technique.

The next section moves to discuss how DS methods perform on different real-world datasets using a filter feature selection approach to further evaluate their performance.

3.5 DS on real world datasets using a filter feature selection

In this section, the focus is to evaluate how DS methods perform on high-dimensional small-instance datasets and compare this to majority voting which is the simplest MCS method. Despite the large number of papers published in DS, there was no comprehensive study available verifying the use of this methods on this specific type of dataset until the work done by Maciel-Guerra *et al.* (2020) [9] which is shown here.

3.5.1 Experimental Methodology

The experiments are conducted on 10 real-world high-dimensional datasets (Table 3.1. Nine of those datasets are obtained from the *Feature Selection datasets* (Arizona State University [92]) and another from the UCI machine learning repository [93]. We considered only datasets with small sample sizes.

All techniques are implemented using the *scikit-learn* [95] and the *DESlib* [96] libraries in Python. The experiments are conducted using 30 replicates. For each replicate, the datasets are randomly divided in 50% for the training set, 25% for the Region of Competence set and 25% for the test set as suggested by Cruz *et al.* [4]. These divisions are performed preserving the proportion of samples for each class by using the stratified k-fold cross validation function in the *scikit-learn* [95] library.

The pool of classifiers is composed of 11 decision trees, as suggested by Woloszynski *et al.* [67], with pruning level set to 10. The pool is generated using the bagging technique, similarly to the methodology followed by Woloszynski in [29, 67]. An odd number of classifiers is chosen to overcome decision ties. These classifiers are used due to their instability when trained with different sets of data, i.e., small differences on the training set can create different trees [67]. Following the recent survey on DS techniques [4], the size of the Region of Competence K is set to 7 neighbours for all the techniques based on k-NN. Moreover, as suggested by Cruz and Soares in [4, 63, 64], 30% of the base classifiers are selected using accuracy and diversity for the techniques DES-kNN and DES-kMeans.

In addition, the number of clusters of DES-kMeans is set to 5.

The Friedman test F_F with Iman-Davenport correction [97] is employed for statistical comparison of multiple classifier system techniques as suggested by Cruz and Demsar in [4, 98]. The rank of each method is calculated using the weighted ranking approach proposed by Yu in [99], which considers the differences among the average performance metric values between classifiers for each dataset [99]. The best performing algorithm is the one with the lowest average rank. Next, as suggested by [98], to compare all classifiers against a control, we use the Bonferroni-Dunn test with the following test equation to compare two classifiers:

$$z = (R_i - R_j) \Big/ \sqrt{\frac{k(k+1)}{6N}} \tag{3.1}$$

where R_i is the rank of i-th classifier, k is the number of classifiers and N is the number of datasets. The z value is than used to find the corresponding p-value from the two-tailed normal distribution table, which is subsequently compared to an appropriate significance level α. The Bonferroni-Dunn test subsequently divides α by $k-1$ to control the family-wise error rate. The level of $\alpha = 0.05$ is considered as significance level. Hence, the level of $p < 0.0022$ was considered as statistically significant.

3.5.2 Results

Accuracy is calculated for all experiments and averaged over the 10 replications. In addition, the rank of all classifiers for each dataset is calculated according to the weighted ranking approach proposed by Yu in [99] and averaged to measure the Z-score to find its respective p-value. With 22 classifiers and 10 datasets, the Friedman test is distributed according to the F distribution with $22 - 1 = 21$ and $(10 - 1) \times (22 - 1) = 189$ degrees of freedom. The critical value of F(21,189) for $\alpha = 0.0001$ is 2.8165.

The first experiment assesses classifier performance without feature selection. Table 3.3 shows the average accuracy and standard deviation for each dataset, the average rank, Z-score and p-value results for all the classifiers that had a rank lower than majority voting without feature selection. The F_F statistic is 4.7468, so the null-hypothesis can be rejected with 99.99% confidence. To compare all classifiers against a control, majority voting, the Bonferroni-Dunn test is used to measure the Z-score for each classifier. Even though there are 3 classifiers (KNORA-U, KNOP-U and DES-P) with a better rank than majority voting, none of them is statistically different from majority voting.

Table 3.3: Average accuracy, ranking, z-score and respective p-value for the classifiers that had a lower rank when compared with majority without feature selection and the oracle results

	knop u	knora u	des p	majority voting	oracle
Allaml	0.9333 ± 0.0563	0.9296 ± 0.0573	0.9296 ± 0.0573	0.9278 ± 0.0576	1 ± 0
Arcene	0.7067 ± 0.0646	0.7173 ± 0.0667	0.704 ± 0.0576	0.71 ± 0.0586	0.996 ± 0.0095
Basehock	0.9045 ± 0.0138	0.8922 ± 0.0132	0.8923 ± 0.0132	0.8917 ± 0.0128	0.9625 ± 0.013
Colon	0.7542 ± 0.0926	0.7604 ± 0.0983	0.7417 ± 0.1067	0.7438 ± 0.0932	0.9896 ± 0.0233
Dexter	0.8789 ± 0.0357	0.8722 ± 0.0366	0.8731 ± 0.0368	0.8729 ± 0.0367	0.992 ± 0.0111
Gli	0.8136 ± 0.0678	0.8212 ± 0.0713	0.8273 ± 0.0737	0.8152 ± 0.0713	0.9924 ± 0.0169
Pcmac	0.8648 ± 0.0162	0.8582 ± 0.0158	0.8576 ± 0.016	0.8575 ± 0.016	0.9421 ± 0.0261
Prostate	0.8782 ± 0.0724	0.8833 ± 0.064	0.8821 ± 0.062	0.8821 ± 0.0688	0.9936 ± 0.0143
Relathe	0.825 ± 0.0205	0.8085 ± 0.0226	0.8121 ± 0.0228	0.8076 ± 0.0217	0.9525 ± 0.0193
Smkcan	0.6298 ± 0.0637	0.6255 ± 0.0551	0.6262 ± 0.0661	0.6135 ± 0.053	0.9986 ± 0.0053
Rank	5,60	6,49	6,82	7,47	-
z score	0,6413	0,3374	0,2220	0	-
p-value	0,5213	0,7358	0,8243	1	-

The second experiment (Table 3.4) employs the univariate feature selection method. Instead of selecting a specific number of features, a p-value is computed using the ANOVA F-test and a family wise error rate is used to select them with a 95% confidence level. For high-dimensional datasets it is necessary to compute a feature selection method to reduce the complexity of the problem. Nonetheless, this is not an easy task due to the "curse of dimensionality". Therefore, the feature selection method chosen must be fast to compute because of the large number of features. This is the reasoning for choosing a filter method as the feature selection approach. For this experiment, the F_F statistical value was 5.6171. Aposteriori, KNORA-U and KNOP-U had a lower rank when compared with majority voting, nevertheless, these ranks are not statistically different.

The aforementioned results show that for all the datasets we tested with more features than samples dynamic selection methods are statistically equivalent to a simple method such as majority voting. This result differs from the recent reviews in the literature [4, 6] that showcased the higher performance of DS methods over majority voting on low-dimensions datasets. Nonetheless, the filter feature selection method chosen was able to reduce drastically the number of features (Table 3.5) and increase the performance of

Table 3.4: Average accuracy, ranking, z-score and respective p-value for the classifiers that had a lower rank when compared with majority with univariate feature selection based on the ANOVA-F test with Family-wise Error rate and the oracle results

	aposteriori	knop u	knora u	majority voting	oracle
Allaml	0.9111 ± 0.0682	0.9315 ± 0.0652	0.9333 ± 0.0664	0.9333 ± 0.0664	1 ± 0
Arcene	0.6907 ± 0.0593	0.7727 ± 0.065	0.7693 ± 0.0655	0.766 ± 0.0687	0.9913 ± 0.0123
Basehock	0.9048 ± 0.0144	0.9063 ± 0.0128	0.895 ± 0.0137	0.8929 ± 0.0136	0.9633 ± 0.0125
Colon	0.8625 ± 0.0987	0.7896 ± 0.0876	0.7896 ± 0.0876	0.7938 ± 0.0886	0.9688 ± 0.0419
Dexter	0.8949 ± 0.0262	0.9009 ± 0.0174	0.8976 ± 0.0166	0.8962 ± 0.0204	0.99 ± 0.0089
Gli	0.8864 ± 0.0721	0.8515 ± 0.0778	0.8545 ± 0.0866	0.8606 ± 0.0813	0.9924 ± 0.0169
Pcmac	0.8737 ± 0.0138	0.8684 ± 0.0186	0.8666 ± 0.0184	0.8641 ± 0.0153	0.9198 ± 0.0368
Prostate	0.8949 ± 0.0432	0.8936 ± 0.0656	0.8885 ± 0.0669	0.8897 ± 0.0657	0.9885 ± 0.0202
Relathe	0.8313 ± 0.0166	0.8274 ± 0.0204	0.8183 ± 0.0209	0.8139 ± 0.0204	0.9198 ± 0.0374
Smkcan	0.7553 ± 0.0568	0.7333 ± 0.0688	0.7369 ± 0.0685	0.7355 ± 0.074	0.9872 ± 0.0224
Rank	5,89	5,9	7,3	7,81	-
z score	0,6606	0,6585	0,1756	0	-
p-value	0,5089	0,5102	0,8606	1	-

most classifiers over all datasets.

The type of datasets used in our work might explain the reasons of our findings. The datasets investigated have a far larger number of features compared to the number of instances. This situation poses a problem for machine learning techniques for some reasons: (1) wrapper methods require a reasonable computational time to select a subset of features in a large search space, hence the selection of a filter technique to reduce the dimensionality; (2) it is likely that there is insufficient data to cover the entire feature space, because the reduction of dimensionality increased the performance of 97% of 22 classifiers over 10 datasets; (3) Euclidean distance does not work on high-dimensional spaces since points are equally distance from one another.

We focused on demonstrating that DS methods did not have high performance levels on datasets with high-dimensionality and low sample sizes when compared with a simple

Table 3.5: Number of features after applying the filter univariate feature selection based on the ANOVA-F test with Family-wise Error rate

datasets	Features before Filter	Features after Filter	Reduction
Allaml	7129	130	98,18%
Arcene	10000	937	90,63%
Basehock	4862	286	94,12%
Colon	2000	16	99,20%
Dexter	20000	36	99,82%
Gli	22283	265	98,81%
Pcmac	3289	59	98,21%
Prostate	5966	198	96,68%
Relathe	4322	126	97,08%
Smkcan	19993	63	99,68%

MCS method such as majority voting. The results suggest that the Euclidean distance used by most of the methods is not working and therefore an alternative must be proposed for these types of dataset. Moreover, feature selection could be incorporate to the DS framework to select the most important features for each sample. Although the results suggest an increase in performance, they are still far from the oracle. This indicates that the features selected might still not be the best subset.

In addition, due to the properties of high-dimensional spaces, clusters can be masked [10]; and a phenomena called *local feature relevance* happens, i.e., different subsets of features are relevant for different clusters [39]. This might explain the reason why the accuracy after feature selection was still further apart from the oracle and further investigations must be conducted to overcome this issue and improve even further the results.

3.5.3 Conclusions

In this section, we investigated how DS methods perform on high-dimensional datasets, more specifically those with a sample-feature ratio below one. We compared 21 DS methods against the majority voting method. Our approach used the Friedman test with the Iman-Davenport correction to compare the averaged weighted ranking of each classifier for all datasets. If the null-hypothesis is rejected, the Bonferroni-Dunn test is used as a post-hoc test to compare all classifiers against a control (majority voting). Experiments with and without feature selection were performed and showed that for high-dimensional datasets the DS methods are statistically equivalent to the majority voting. For both studies , with and without feature selection, the null-hypothesis of the F_F statistic was reject with a confidence of 99.99%. Moreover, in both studies, the Bonferroni-Dunn test showed that none of the best ranked classifiers are statistically different from the majority voting classifier, which contradicts most of the results in the literature. These results indicate that modifications to the traditional DS framework could be beneficial.

The next section discusses how DS methods compare with single classifiers before and after a wrapper feature selection approach using a protein microarray dataset.

3.6 DS on real-word data using a wrapper approach

In this section, the focus is to evaluate and compare single classifiers with DS methods using a wrapper feature selection approach. Since we have to train and cross-validate our model for each feature subset combination, this approach is much more expensive than a filter approach. When compared to filter methods, a wrapper approach tends to find features better suited to the predetermined learning algorithm resulting in superior learning performance, but it also tends to be more computationally expensive which increases with the number of features in the dataset. Since, the datasets studied on the previous section have a high number of features, we decided to test this approach on a protein microarray dataset which has less samples than features, but with a feasible number of features for a wrapper approach. Here we used a wrapper backward selection (WBS) with a RELM as the embedded model.

Protein microarrays are a powerful tool employed in allergy diagnostics, as it monitors interactions between the immune system and allergens. In microarray data, there is information regarding the fluorescence of binding signals, which are proportional to the concentration of an antibody's reaction to each spot containing allergens in the microarray. As healthy and unhealthy animals are expected to mount different immune responses to allergens, the analysis of existing microarray data should enable the determination of prediction models for early diagnosis of allergies. Another important aspect of the study of microarray data is that it generally carries a significant number of irrelevant features leading to miss classification. The determination and pruning of those irrelevant features tend to promote performance improvement.

Nine state-of-the-art classification methods (Logistic Regression [100], Linear and Non-linear Support Vector Machines (SVM) [56, 57], Random Forest [53], Multi-Layer Perceptron (MLP) Neural Networks [54, 55], AdaBoost [101], Naive Bayes [102], Linear Discriminant Analysis (LDA) [103, 104] and Regularized Extreme Learning Machine (RELM) [81, 105]) were compared with 21 DS methods (Table 2.1).

3.6.1 The Insect Bite Hypersensitivity dataset

A total of 196 horses comprising 49 non-affected (healthy) controls and 147 IBH-affected horses are included in the study. A complex protein microarray containing 384 extracts and pure proteins from a wide range of protein families (e.g. fruit, dairy, seeds, pollen, fungi, insects, fish) is assembled essentially as described by Marti *et al.* (2015) [106]. The dataset does not contain missing values and is pre-processed according to the scheme described by Vigh-Conrad *et al.* (2010) [107]. The authors normalise the data by correcting the autofluorescence in both red and green channels [107]. They assume that for each spot, the red channel intensity (R) is the sum of the fluorescence of the second antibody - IgE (R_{IgE}) and autofluorescence (R_{AF}); while the green channel intensity (G), since is not affect by the second antibody, is, therefore, equal to its autofluorescence (G_{AF}). On slides with buffer only, they observed that $R_{AF} = mG_{AF} + b$, in other words, a linear relationship exists between the red and green channels [107]. R_{AF} and G_{AF} were, therefore, obtained by applying linear models for each allergen separately, and the resulting value of R_{AF} was subtracted from R to obtain R_{IgE}. By using this normalisation, the final intensities are centered at 0. Finally, the data is further normalised for each feature to have a range between 0 and 1.

3.6.2 Experimental Methodology

The same dataset employed by Marti *et al.* [106] to study equine insect bite hypersensitivity (IBH) is adopted in this study. The dataset contains 109 observations (66 healthy controls and 43 IBH diseased animals) described by 193 features. The minimum value of this dataset is 0 and the maximum value is 874.91. The dataset does not contain missing values. The dataset is pre-processed according to the scheme adopted by Marti *et al.* [106], in which the negative control microarray data (consisting of all reagents except the animal serum) was subtracted from the sample slide to eliminate non-specific binding and inherent autofluorescence of some proteins; after the occurrence of this process, the slides received a second normalization (Equation 3.2, where n_x is the norm of a 1 by N

vector **x**), involving the sum of absolute values of all expressions (associated with each individual), in order to reduce technical variability. Finally, the data is mean centered and scaled to unit standard deviation for each feature.

$$n_x = \sum_{i=1}^{N} |x_i| \tag{3.2}$$

For evaluation of the results we employ classification area under the receiver operating characteristic curve (AUC) of each classifier.

The performance of the classifiers Naïve Bayes [102], Linear and Non-Linear (RBF kernel) Support Vector Machines (SVM) [56, 57], Random Forest [53], Multi-Layer Perceptron Neural Networks (MLP) [54, 55], AdaBoost [101], Logistic Regression and Linear Discriminant Analysis (LDA) [103, 104] was investigated using the *scikit-learn* library in Python [95]. The Regularized Extreme Learning Machine classifier was implemented in MATLAB R2016a (The MathWorks, Inc., Natick, Massachusetts, United States), using the proposal of Kulaif and Von Zuben [82].

For the classifiers, the following set of values are employed for the hyper-parameters, before and after feature selection:

- **Logistic Regression:** inverse of regularization strength C = [0.001, 0.01, 0.1, 1, 10, 100, 1000].
- **Linear SVM:** penalty parameter of the hinge loss error C = [0.001, 0.01, 0.1, 1, 10, 100, 1000].
- **Random Forest and Adaboost:** Number of estimators = [2, 4, 8, 16, 32, 64, 128, 256, 512, 1024].
- **MLP Neural Network:** α (L2 penalty parameter) = [0.001, 0.01, 0.1, 1, 10, 100], learning rate (initial learning rate used to control the step size in updating the weights with adam solver) = [0.001, 0.01, 0.1, 1] and hidden layer sizes = [10, 20, 40, 100, 200, 300, 400, 500].
- **Non-linear SVM with RBF kernel:** γ (RBF kernel coefficient) = [0.0001, 0.001,

0.01, 0.1] and C (L2 penalty parameter) = [0.001, 0.01, 0.1, 1, 10, 100, 1000].

- **RELM:** λ (L2 penalty parameter) = [0.0001, 0.001, 0.01, 0.1, 0, 1, 10, 100, 1000], hidden layer sizes = [200, 201, 202, ..., 698, 699, 700], and random weights of the hidden layer in the range [-0.5, 0.5].
- **Naive Bayes and LDA:** do not have hyper-parameters.

For the WBS approach, an RELM with λ (L2 penalty parameter) = [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000] and hidden layer size of 500 was adopted.

3.6.3 Results

Table 3.6 shows the accuracy results, before and after feature selection. Each experiment is conducted thirty times. The numbers after the "±" symbol are standard deviation.

The wrapper feature selection method implemented as the WBS with RELM as the embedded model produced 36 features which are considered the most relevant. The third column of Table 3.6 shows that the feature selection is able to remove redundant and non-important proteins for the classification of IBH for most classifiers, since their performance is improved after feature selection or remains the same.

Comparing the accuracy before and after feature selection, LDA has the highest increase (36.82%), followed by Linear SVM with an increase of 26.53%. No classifiers had an accuracy over 0.9 before feature selection, while 4 methods (Logistic Regression, Linear SVM, RBF SVM and LDA) achieve an accuracy over 0.9 after feature selection. Logistic Regression and SVM with RBF kernel have improvements of 16.74% and 14.96%. Although the literature shows that DS methods usually have a higher performance when compared with single classifiers or other ensemble techniques, for this specific dataset DS under-performed.

These results can potentially be explained by the fact that the IBH microarray data has far more features than samples. As the DS methods employ a bagging algorithm to generate the pool of classifiers in different regions of the feature space, it has fewer in-

Table 3.6: Accuracy Results in Insect bite Hypersensitivity dataset

Classifiers	**Before Feature Selection**	**After Feature Selection**
Logistic Regression	0.813 ± 0.024	0.949 ± 0.014
Linear SVM	0.741 ± 0.035	0.937 ± 0.016
RBF SVM	0.817 ± 0.024	0.939 ± 0.016
Random Forest	0.827 ± 0.021	0.894 ± 0.023
AdaBoost	**0.885 ± 0.021**	0.878 ± 0.017
Decision Tree	0.81 ± 0.032	0.818 ± 0.026
Naive Bayes	0.698 ± 0.021	0.767 ± 0.019
QDA	0.546 ± 0.042	0.606 ± 0
LDA	0.713 ± 0.036	**0.975 ± 0.01**
CR	0.829 ± 0.065	0.771 ± 0.068
MCR	0.793 ± 0.061	0.796 ± 0.068
OLA	0.839 ± 0.062	0.836 ± 0.056
LCA	0.761 ± 0.052	0.786 ± 0.066
RELM	0.803 ± 0.012	0.864 ± 0.031
Apriori	0.818 ± 0.069	0.811 ± 0.045
Aposteriori	0.725 ± 0.061	0.782 ± 0.095
MLA	0.832 ± 0.046	0.832 ± 0.028
DES-kNN	0.796 ± 0.08	0.804 ± 0.07
DES-kMeans	0.85 ± 0.076	0.814 ± 0.047
KNORA-E	0.857 ± 0.068	0.864 ± 0.045
KNORA-U	0.814 ± 0.079	0.8 ± 0.074
KNOP-E	0.873 ± 0.059	0.861 ± 0.037
KNOP-U	0.811 ± 0.064	0.8 ± 0.058
META-DES	0.876 ± 0.057	0.857 ± 0.06
DSOC	0.861 ± 0.056	0.814 ± 0.061
DES-RRC	0.832 ± 0.094	0.854 ± 0.046
DES-EXP	0.879 ± 0.064	0.857 ± 0.06
DES-KL	0.875 ± 0.058	0.857 ± 0.062
DES-MD	0.814 ± 0.061	0.839 ± 0.056
DES-P	0.875 ± 0.058	0.857 ± 0.062

stances available which do not allow the methods to create effective regions of competence for classification.

3.6.4 Conclusions

Accurate diagnosis of a disease is vital for a successful therapy. Protein microarrays are a powerful tool employed in allergy diagnostics in order to monitor interactions of antibodies with allergens. In this section which indicates the results obtained by Maciel-Guerra *et al.* (2019) [8], we investigated the use of DS methods in microarray data.

We used an insect bite hypersensitivity dataset as our case study and compared the DS results with traditional machine learning methods. We also compared the results of DS with single and static classifiers before and after feature selection.

Machine learning classifiers along with WBE for feature selection were investigated. A Regularized Extreme Learning Machine with WBS was used as a feature selection method. We compared the classification results before and after WBS. The DS methods did not have a higher increase in performance. In addition, most of the outputs of the 21 different DS models produced similar results. Nonetheless, most of single and static classifiers had a higher increase in performance. These results may be explained by the fact that this dataset has more features than samples, which leaves less samples to form the region of competence, i.e., with few samples in a high-dimensional space, the k-NN algorithm may not find the correct regions of competence for each unseen sample.

3.7 Chapter Summary

In this chapter, characteristics of high-dimensional datasets were investigated. In particular, the hubness and distance concentration phenomena were discussed with experiments showcasing the issues and challenges 11 real-world datasets have. Next, experiments were conducted to investigate the challenges dynamic selection methods face when dealing with small instance high-dimensional datasets. First, DS methods were tested on synthetic data with different numbers of features and samples generated based on Trunk's classification problem. The results indicate that all DS methods had a decrease in performance when the number of dimensions increase. Next, DS methods were compared with majority voting and single classifiers on different feature selection approaches. First, the performance of DS methods was tested on 10 real-world datasets and compared with majority voting. A filter feature selection was implemented to evaluate the performance before and after feature selection. The results showed that the performance of majority voting was statistically equivalent in comparison to the DS methods. Second, nine state-of-art classification methods were compared with 21 DS methods before and after the feature selection with a wrapper backward selection in a protein microarray dataset.

The results indicate that single and static classifiers had a higher performance when compared to the DS methods before and after the wrapper feature selection. Overall, these results indicates that the traditional DS framework cannot be used on small instance high-dimensional datasets, since the performance of the k-NN deteriorates as the number of dimensions increases. In addition feature selection approaches, like filter and wrapper methods, can improve the overall classification but they still fail to make the DS methods statistically better than majority voting and single classifiers.

Therefore, the next chapter moves to propose a novel dynamic selection method based on a subspace clustering algorithm to reduce the impacts of hubness and distance concentration in small instance high-dimensional datasets. The subspace clustering method substitute the k-NN in the region of competence.

Chapter 4

Subspace-Based Dynamic Selection (SBDS) Framework

***

4.1 Introduction

In this chapter, we introduce the SBDS framework (Figure 4.1), which aims to have a better performance in high-dimensional, small instance datasets when compared to dynamic selection methods.. To achieve this, we propose to combine subspace clustering and nearest subspace search. The objective is to improve feature relevance, while maintaining or increasing the performance when compared with DS methods. The main difference of SBDS over DS is the use of subspace clustering to search through the feature and sample spaces for relevant clusters (equivalent to regions of competence in DS). By training different classifiers in distinct subspaces, we ensure that each classifier (or a combination of classifiers) is an expert in a different region of the feature space [4–6].

The core aim of this thesis is to provide a framework to evaluate high-dimensional small instance datasets. In order to do this we developed a framework based on dynamic selection with the addition of a subspace clustering method to improve the selection of the features. Moreover, since the subspace clustering method gives the best features for an specific group of samples we are able to further understand their importance. Therefore, SBDS also helps with knowledge discovery which in biological datasets, genomics for instance, can help understand which genes are more important for each sample. Specially, this chapter aims to answer one of our main research questions: **"How can we incorporate subspace clustering methods to the DS framework?"**

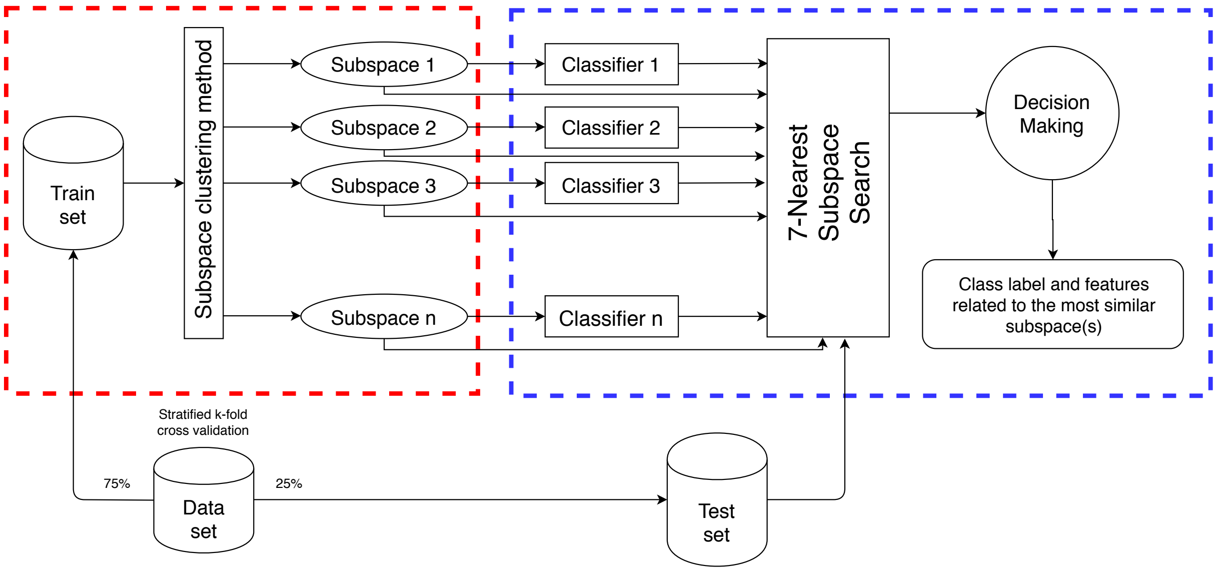

Train set
Subspace clustering method
Subspace 1
Subspace 2
Subspace 3
Subspace n
Classifier 1
Classifier 2
Classifier 3
Classifier n
7-Nearest Subspace Search
Decision Making
Class label and features related to the most similar subspace(s)
Stratified k-fold cross validation
75%
25%
Data set
Test set

Figure 4.1: Proposed SBDS framework. The red dashed square indicates the Step 1 (subspace clustering) and the blue dashed squared indicates Step 2 (k-Nearest Neighbour Search and decision making).

This chapter first introduces an overview over each part of the framework in general terms. The data generation process is explained, followed by how one dimensional clusters are found and how they can be merged to form subspaces. After the merging procedure some insights on how to select the subspaces are given. The chapter then moves to give further details on the two version proposed for the SBDS framework: the SBDS (Section 4.3) and the Classifier SBDS (cSBDS) (Section 4.4). The core of the framework is kept on both version with the main differences being on the following steps:

1. Subspace clustering method
2. Addition of a filter method to improve subspace selection
3. Nearest Subspace Search

These differences are further discussed in sections 4.3 and 4.4 with the comparisons and the rationale behind the differences explained.

4.2 Overview

In this section, we introduce each step of the proposed SBDS framework (Figure 4.1) with the reasoning behind each step.

4.2.1 Generate Data

Data is normalised using a Min-Max approach, which scales the data between 0 and 1 for each feature. Then data is randomly divided into a training set and a testing set using a stratified k-fold cross validation to preserve the proportion of samples for each class.

4.2.2 Find One-Dimensional Clusters

Given a training dataset, we determine the clusters in each dimension using a Gaussian Kernel Density Estimator (GKDE). A cluster is defined by a local maximum of the estimated density function, i.e. all points near the local maximum are assigned to the same cluster (Fig. 4.2).

A GKDE helps identify the density of a distribution of the data, i.e. it helps identify where a group of samples is present and where is not present. Therefore, naturally it can be used to cluster one dimensional data by creating clusters of points near a local maxima (high density), separated by the points in a local minima (low density). For this to happen, GKDE needs 4 steps:

1. normalise data

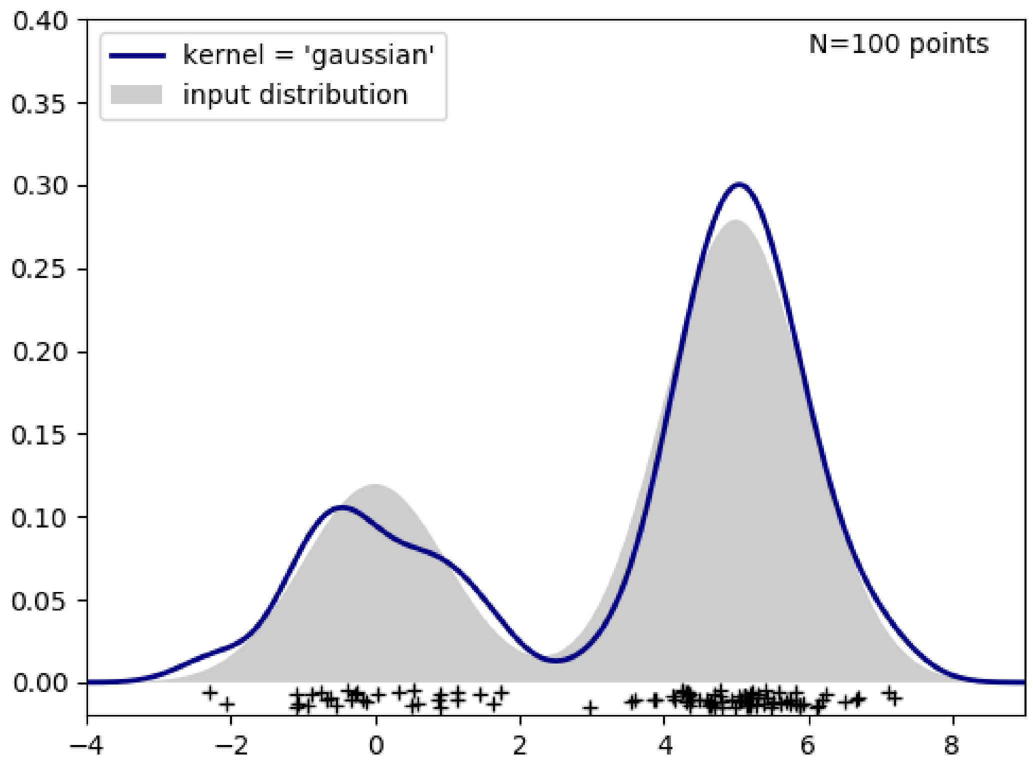


Figure 4.2: Example of a Gaussian Kernel Density Estimator (GKDE) to find one-dimensional clusters. Based on the density distribution of the points local minima and maxima areas are established. The points between two local minima are considered a cluster

2. compute densities

3. find areas of local maxima

4. find areas of local minima

5. the points between two local minima form a cluster

One of the most important aspects to find a suitable density estimator is the choice of bandwidth. A bandwidth very narrow (values close to 0) can lead to a high-variance estimate (over fitting), i.e. the presence or absence of a single point makes a large difference. On the other hand, a too wide bandwidth can lead to a high-bias estimate (under fitting), where the topology of the data is lost. Therefore, a grid search cross-validation is used to empirically optimise the bandwidth which maximises the data log-likelihood using a 20-fold cross validation.

4.2.3 Merging Procedure

After finding all one-dimensional clusters, a merge process is conducted to obtain the subspace clusters. Here we adopt a general method based on Tian and Gu (2019) [87]. The authors proposed to first merge similar clusters with different subspaces by using the Jaccard coefficient. In addition, if a cluster is contained in another cluster (in terms of samples) they must be merged [87]. The summary of the merging process is the same as the one proposed by Tian and Gu (2019) [87] and is described as follows:

Step 1 : set an empty set $\mathcal{D}$

Step 2 : choose one dimension that has not been merged and determine its clusters using a GKDE

Step 3 : choose one cluster of the current dimension

Step 4 : compare the selected cluster with all subspace clusters in $\mathcal{D}$ to find if there is a similar cluster by computing the Jaccard coefficient (Equation 4.1). If no similar subspace exists, go to Step 7

Step 5 : merge the chosen cluster with its similar one

Step 6 : compare the selected cluster with all subspace clusters in $\mathcal{D}$ to find if it is contained in another subspace (Equation 4.2). If it is merge them and go to Step 8

Step 7 : add the selected subspace to $\mathcal{D}$ and compare the selected cluster with all subspaces in $\mathcal{D}$ to find if there is one that contains it. If there is, merge them

Step 8 : if all cluster of current dimension are selected, go to Step 9. Otherwise go to Step 3

Step 9 : if all dimensions are merged, return $\mathcal{D}$. Otherwise go to Step 2.

The Jaccard coefficient to measure the similarity between two subspaces is defined as:

$$J(E,F) = \frac{|E \cap F|}{|E \cup F|} \tag{4.1}$$

where E and F are the samples of two subspaces. The containment relationship of E and F is defined as:

$$\begin{aligned} C_1(E,F) &= \frac{|E \cap F|}{|E|} \\ C_2(E,F) &= \frac{|E \cap F|}{|F|} \end{aligned} \tag{4.2}$$

if $C_1(E,F)$ is close to 1 (in this case the threshold is set at 0.9) and $C_2(E,F)$ have a smaller value (below the set threshold), E is contained in F. On the other hand, if $C_2(E,F)$ is close to 1 (in this case the threshold is set at 0.9) and $C_1(E,F)$ have a smaller value (below the set threshold), F is contained in E.

After the merging procedure we have all the subspaces found for the dataset.

4.2.4 Subspace Selection

For each unknown test sample, we need to determine the nearest or best subspaces to classify our unseen sample. We propose two versions of the SBDS framework which mainly differ in this aspect of the framework. Version 1 is proposed on the work Maciel-Guerra *et al.*, 2020 [42] where a 7-nearest subspace search is performed using a distance metric. Version 2 selects the best subspaces based on the performance of a RBF-SVM classifier.

The next section describes the first version of the SBDS framework in more details. Moreover, it argues the advantages and disadvantages of using this version.

4.3 Subspace-Based Dynamic Selection (SBDS) framework

In this section, we introduce the proposed SBDS framework (Algorithm 1) which aims to merge different concept of DS, subspace clustering and nearest subspace search. The objective of this framework is to improve feature relevance in high-dimensional small-instances datasets while maintaining or increasing the performance when compared with DS methods. The advantage of SBDS over DS is the use of subspace clustering to search through the feature and sample spaces for relevant clusters (equivalent to regions of competence in DS). By training different classifiers in distinct subspaces, we ensure that each classifier (or a combination of classifiers) is an expert in a different region of the feature space [4–6]. Moreover, in practical problems, different instances have different classification difficulties and may be located in different subspace clusters. Hence, our hypothesis is that adopting different subspaces to predict different test samples may increase the performance of a multiple classifier system [4, 48].

Therefore, the first version of the SBDS framework consists of finding possible clusters for each individual dimension using a GKDE approach which uses all samples regardless of their class. The example given in Figure 4.2 indicates how the one-dimensional cluster is selected using all the samples. Subsequently, a merging process is conducted to combine the one-dimensional clusters to form the subspaces. After the merging process, we train one k-NN classifier per subspace cluster, iff the subspace contains samples for more than one class. Otherwise, if the subspace contains just one class, we don't train a k-NN on it. The next step is to find the nearest subspaces to each test sample in order to make their prediction.

For each unknown test sample, we need to determine the 7-nearest subspaces. The number of subspaces is the same as that used in most papers in DS to define the size of the region of competence. This step measures the similarity between a point and a subspace.

We first calculate the centroid $\mathcal{C}_i$ for each subspace $\mathcal{S}^i$, by calculating the average for each feature for all instances in $\mathcal{S}^i$. Subsequently, we measure the average Euclidean distance (d_{Sc}) between all points in $\mathcal{S}^i$ and $\mathcal{C}_i$. We also calculate the Euclidean distance (d_{Tc}) between the test sample $\mathcal{Q}$ (using only the dimensions within $\mathcal{S}^i$) and $\mathcal{C}_i$.

The ratio between the two distances d_{Tc} and d_{Sc} is calculated to verify whether the instance $\mathcal{Q}$ belongs to the subspace. However, we observed that this ratio does not remain constant as the dimensionality increases. High-dimensional data produces higher distance values, which adds bias toward the k-nearest low dimensional subspaces, since we are selecting the k smallest ratios. To prevent this bias the ratio needs to be multiplied by a function of the dimension, in a way that the ratio between $\mathcal{Q}$ and all subspaces are comparable.

Equation 4.3 therefore gives the final value used to compare the point-to-subspace similarities. The multiplier factor $\frac{1}{1+\sqrt{(dim)}*\ln(dim)}$ was found empirically for the IBH dataset. Finally, the 7 smallest ratios are selected.

$$\mathcal{R} = \frac{d_{Tc}}{d_{Sc}} * \frac{1}{1+\sqrt{dim}*\ln(dim)} \tag{4.3}$$

where dim is the dimension of the subspace.

The predictions are given by the 7 classifiers (if more than one class is present) or the original class (if a single class is present) associated with each one of the 7-nearest subspaces, and a majority voting is used to define the label of the test sample.

4.3.1 Experimental Methodology

For evaluating the results we employ accuracy, sensitivity and specificity for each classifier. The experiment is carried out using 30 replications. For each replication, the datasets are randomly divided as 75% for training and 25% for testing. These divisions are performed preserving the proportion of samples for each class by using the stratified k-fold cross validation function in the *scikit-learn* [95] library.

The same DS methods used are listed in Table 2.1. More information about each method can be found on their respective reference and on the Appendix A. Similarly to Maciel-Guerra *et al.* (2019) [8], 11 decision trees are used to compose the pool of

Algorithm 1: Subspace-Based Dynamic Selection (SBDS)

```
input  : dataset $\mathbb{D}$
output: $\upsilon$
/* $\upsilon$ is the array with the predicted label for all samples in $\mathbb{L}$ */
1  Split the data into training $\mathbb{T}$ and testing $\mathbb{L}$ ;
2  foreach feature $\in \mathbb{D}$ do
3      Use a GKDE approach to define the one-dimensional cluster;
4      Apply a merging procedure to create the subspaces
5  end
6  foreach $\mathbf{x}_{test} \in \mathbb{L}$ do
7      Calculate the centroid $\mathcal{C}_i$ for each subspace $\mathcal{S}^i$, by averaging each feature for
        all instances in $\mathcal{S}^i$ ;
8      Measure the average Euclidean distance ($d_{Sc}$) between all points in $\mathcal{S}^i$ and $\mathcal{C}_i$ ;
9      Calculate the Euclidean distance ($d_{Tc}$) between the test sample $\mathcal{Q}$ (using only
        the dimensions within $\mathcal{S}^i$) and $\mathcal{C}_i$ ;
10     Calculate the ratio between the two distances $d_{Tc}$ and $d_{Sc}$ and apply a
        multiplier factor $\frac{1}{1+\sqrt{(dim)}*\ln(dim)}$;
11     Select the 7-nearest subspaces; Predict the label of $\mathbf{x}_{test}$ using the 7 selected
        subspaces;
12 end
```

classifiers. The size of the region of competence is set to 7 for all techniques based on k-NN.

4.3.2 Results

In our experiment, SBDS is compared with some state-of-art machine learning methods and some of the most import DS methods in the literature using a protein microarray dataset. Table 4.1 shows accuracy, sensitivity and specificity results for all techniques mentioned on Table 2.1. The numbers after the "±" symbol are standard deviation.

From the obtained results in Table 4.1 it is relevant to observe that all methods learned better the majority class. Moreover, our proposed SBDS framework performed better in terms of accuracy (0.7986) than all DS methods investigated. *A posteriori* had the highest sensitivity (0.9811) and DES-kMeans the highest specificity (0.5166).

These results in a single dataset indicate that SBDS is able to achieve similar results by having an embedded subspace clustering method. This allows us to verify which features were selected in each subspace to predict the label of the unknown test sample.

Table 4.1: Accuracy, sensitivity and specificity results of the IBH dataset

Classifiers	**Accuracy**	**Sensitivity**	**Specificity**
SBDS	**0.7986 ± 0.0455**	0.9270 ± 0.0389	0.4027 ± 0.1199
CR	0.7217 ± 0.0659	0.8171 ± 0.0747	0.4277 ± 0.1133
MCR	0.7612 ± 0.0522	0.8936 ± 0.0512	0.3527 ± 0.1605
OLA	0.7245 ± 0.0493	0.8405 ± 0.0551	0.3666 ± 0.1151
LCA	0.7177 ± 0.0582	0.8288 ± 0.0765	0.3750 ± 0.1268
A Priori	0.7231 ± 0.0631	0.8234 ± 0.0727	0.4138 ± 0.1317
A Posteriori	0.7599 ± 0.0222	**0.9811 ± 0.0253**	0.0777 ± 0.0984
MCB	0.7265 ± 0.0684	0.8315 ± 0.0884	0.4027 ± 0.1219
MLA	0.6844 ± 0.0474	0.7801 ± 0.0623	0.4055 ± 0.1268
DES-kMeans	0.7299 ± 0.0547	0.7990 ± 0.0661	**0.5166 ± 0.1280**
DES-kNN	0.7442 ± 0.0473	0.8676 ± 0.0546	0.3638 ± 0.1402
KNORA-E	0.7361 ± 0.0666	0.8180 ± 0.0754	0.4833 ± 0.1298
KNORA-U	0.7803 ± 0.0437	0.9234 ± 0.0503	0.3388 ± 0.1234
DES-EXP	0.7578 ± 0.0598	0.8585 ± 0.0867	0.4472 ± 0.1603
DES-RRC	0.7768 ± 0.0477	0.9189 ± 0.0563	0.3388 ± 0.1359
DES-MD	0.7578 ± 0.0598	0.8585 ± 0.0867	0.4472 ± 0.1603
DES-KL	0.7626 ± 0.0607	0.9009 ± 0.0772	0.3361 ± 0.1352
DES-P	0.7782 ± 0.0389	0.9072 ± 0.0472	0.3805 ± 0.1069
KNOP-E	0.7211 ± 0.0639	0.8171 ± 0.0757	0.4250 ± 0.1125
KNOP-U	0.7823 ± 0.0429	0.9351 ± 0.0455	0.3111 ± 0.1137
Meta-DES	0.7401 ± 0.0582	0.8387 ± 0.0727	0.4361 ± 0.1192
DSOC	0.7694 ± 0.0508	0.9045 ± 0.0566	0.3527 ± 0.1547

Therefore, our method poses an advantage in comparison to DS methods in terms of giving this additional information of which are the most important features for each sample.

Table 4.2 shows which features were most selected by the 7-Nearest Subspace Search over all test samples in the 30 iterations. The 16 out of the 21 proteins are related to the *Culicoides sp.* allergome family that are clinically the cause of IBH in horses.

4.3.3 Discussion

To overcome the issue of using a k-NN in high-dimensional feature spaces, this thesis suggests the use of subspace clustering. The first version of the Subspace-Based Dynamic Selection (SBDS) framework [42] is created based on this idea, and the incorporation of a subspace clustering method attempts to narrow the search of important features using a specific set of samples. Moreover, by using a subspace clustering technique, SBDS naturally gives the importance of each feature in relation to the prediction of each

Table 4.2: Frequency of proteins that were selected in the 7-Nearest Subspace Search

Allergome name	**Latin name**	**Appearance**
Api g [Root]	Apium graveolens	81.70%
Cul n 10.03	Culicoides nubeculosus	78.88%
Cul o2P	Culicoides obsoletus	77.78%
Mal d [Fruit]	Malus domestica	77.78%
Cul o 7	Culicoides nubeculosus	76.98%
Cul o 7	Culicoides nubeculosus	73.89%
Cul o 7	Culicoides nubeculosus	71.40%
Culicidae Cul E	culicidae	71.24%
C0145	Culicoides obsoletus	64.07%
Cul ob 8	Culicoides obsoletus	61.81%
Mus xp	Musa x paradisiaca	60.80%
Pru p 3	Prunus persica	57.96%
Bos d 4	Bos domesticus	57.93%
Cul o1P	Culicoides nubeculosus	57.64%
Cul o 7	Culicoides nubeculosus	56.63%
Cul n 4	Culicoides nubeculosus	56.05%
Cor a 9	Corylus avellana	55.65%
Culicidae Cul C	culicidae	54.18%
Cul o 1	Culicoides nubeculosus	53.64%
Culicidae Cul D	culicidae	53.04%
Cul o 4	Culicoides obsoletus	50.80%

unknown test samples. Thus making it a more explainable technique when compared to complex DS methods. Nonetheless, the proposed SBDS framework [42] has a disadvantage of relying in a empirical multiplier factor to find the nearest subspaces of an unknown test sample. This multiplier factor was initially introduced to make distances based on different dimensions comparable.

The SBDS framework presented has two limitations based on the dataset investigated: (1) the subspaces were found by using samples from both classes on the dataset; (2) the framework uses a multiplier factor that was found empirically for this dataset in the distance calculation to determine which is the closest subspace to the unknown test sample. By overcoming these limitations we hypothesise that the performance of the SBDS framework could be improved in comparison to DS methods.

The first limitation is important because some subspaces could be lost since the one-dimensional cluster depends on the Gaussian kernel density estimator approach. In some cases, if the points from different classes are not clearly separated, the approach proposed for the SBDS framework will disregard this dimension. However, when analysing the classes separately the differences between these points could be accounted for. In addition, the second limitation is important because the multiplier factor was determined for the IBH dataset, and may not be the correct one for other datasets.

Nonetheless, SBDS achieved the highest accuracy over all DS methods, but did not achieve the highest sensitivity and specificity. Moreover, it has the advantage of showing which features were selected for the classification of each test sample. This is important for accurate diagnosis of a disease and it is vital for a successful treatment. Moreover, precision diagnostic of each separate individual allows doctors to give personalised treatment. The proposed SBDS algorithm brings this possibility by incorporating a subspace clustering method into the DS framework.

The next section presents the modified version of the SBDS framework and compare the two approaches with other high-dimensional datasets. Also, it describes how it is possible to improve the nearest subspace search by investigating different approaches to select the subspaces and if changes in the way subspaces are generated can improve the performance.

4.4 Classifier SBDS (cSBDS) framework

In this section, we introduce a modified version of the SBDS framework to tackle high-dimensional problems in which there is no multiplier factor, making it more general to be used in different datasets. Moreover, modifications were made on the subspace finding algorithm. The focus of this section is to compare the modified framework with the one presented in the previous section and with traditional DS methods using high-dimensional datasets. This section attempts to answer the following research questions:

1. How does the cSBDS framework performs in high-dimensional small instance dataset?

2. Can the proposed cSBDS framework outperform the original SBDS framework?

3. Can the modified SBDS framework select less features?

4. How does the performance of the cSBDS framework compare with traditional DS methods and majority voting?

To answer these questions, 10 real-world datasets with a high number of features and a low number of samples are selected. Three datasets are text based while seven are biomedical datasets all related to different types of cancer (lung, prostate, leukemia, colon, glioma and ovarian).

The main differences between the proposed cSBDS framework (Algorithm 2) when compared to one proposed in the previous section are:

1. The subspaces are selected per class using the GKDE one-dimensional clustering approach.
2. A filter to select previously the most relevant subspaces
3. The removal of the distance metric to find the closest subspaces
4. The incorporation of a RBF-SVM trained in each subspace in the k-Nearest Subspace Search step.

Further details of each stage of the cSBDS framework are described below:

Generate Data In the cSBDS framework the data is randomly divided into 50% for the training set, 25% for the validation set and 25% for the testing set using a stratified k-fold cross validation to preserve the proportion of samples for each class.

Find One-Dimensional Clusters Given a training dataset, we determine the clusters in each dimension using a GKDE. Differently from the original SBDS framework, in which the GKDE is used in the whole set of the training samples; the cSBDS framework determine the clusters using the samples from each class separately. Figure 4.3 indicates the difference between the proposed approach on the SBDS and the one proposed here. This example shows that for this feature if we used all the samples regardless of their class, no clusters would have been found. Nonetheless, by having the GKDE done on each class it is possible now to identify two clusters for this feature.

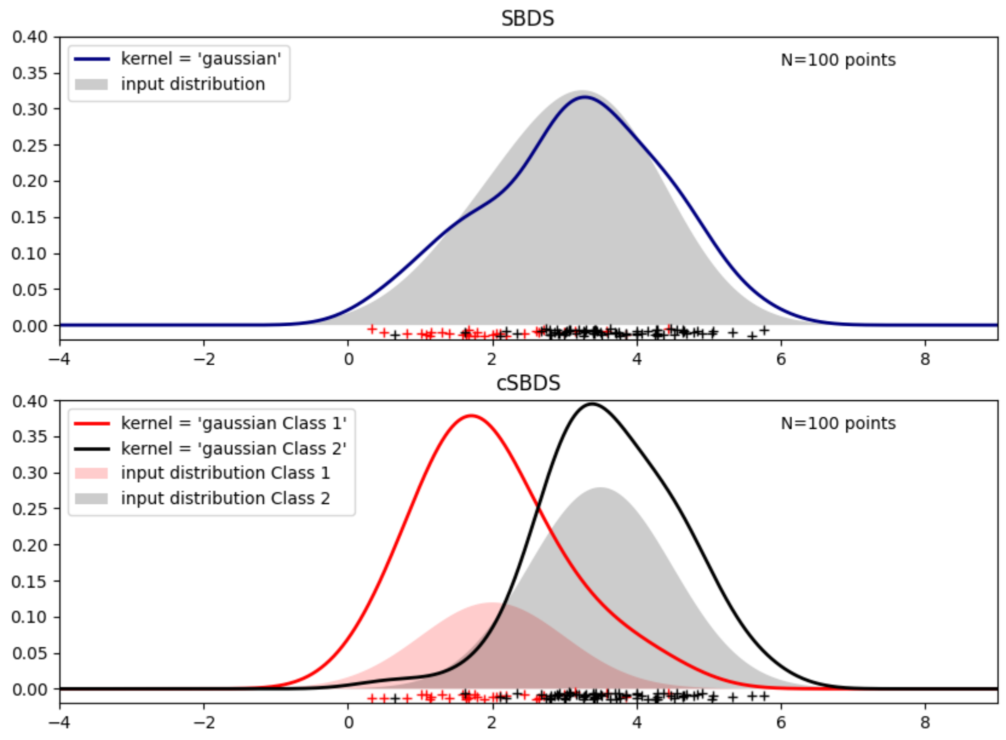


Figure 4.3: Comparison between the Gaussian Kernel Density Estimator (GKDE) approach on the SBDS and the cSBDS to find one-dimensional clusters.

Merging Procedure After all one-dimensional clusters, a merge process is conducted to obtain the subspace clusters. The merging procedure is conduct for each class individually, creating subspaces with only one class. Here we adopt a general method based on Tian and Gu (2019) [87] and further described on Section 4.2.3.

k-Nearest Subspace Search After the merging process, all the one-dimensional subspaces are removed. Next, one RBF-SVM classifier was trained per subspace and evaluated using the validation set. Since each subspace is formed by a singles class, the classifier is trained on the entire set of the training samples, but only on the features for each subspace. A grid search for the gamma kernel coefficient $(0.0001, 0.001, 0.01, 0.1)$ and the regularisation parameter $(0.001, 0.01, 0.1, 1, 10, 100, 1000)$ was used to identify the best hyper-parameters. A filter approach was used to remove the subspaces that did not achieve a threshold *delta* of performance in terms of accuracy, which is reduced by 5% iff the number of subspaces selected is below

7. For each unknown test sample, the RBF-SVM classifiers trained in the previous steps are used to predict the class of each unknown test sample. Majority voting is used to determine the final class and the subspaces used to determine it.

Algorithm 2: Classifier SBDS (cSBDS)

```
input  : dataset $\mathbb{D}$
output: $v$
/* $v$ is the array with the predicted label for all samples in $\mathbb{L}$ */
1  Split the data into training $\mathbb{T}$, validation $\mathbb{V}$ and testing $\mathbb{L}$ ;
2  foreach feature $\in \mathbb{D}$ do
3      Use a GKDE approach to define the one-dimensional cluster using each class
         separately;
4      Apply a merging procedure to create the subspaces ;
5      Remove the 1-dimensional subspaces ;
6      Train one RBF-SVM classifier in each subspace using the entire set of the
         training samples ;
7      Select the best hyper-parameters and evaluate using the validation set ;
8  end
9  foreach $\mathbf{x}_{test} \in \mathbb{L}$ do
10     Filter the subspaces using the validation set that did not achieve a threshold
         $delta$ of performance in terms of accuracy, which is reduced by 5% $iff$ the
         number of subspaces selected is below 7. ;
11     Predict the label of $\mathbf{x}_{test}$ using the RBF-SVM ;
12     Use a majority voting to determine the label of $\mathbf{x}_{test}$ and output only the
         subspaces that predicted that label ;
13 end
```

4.4.1 Experimental Methodology

All techniques are implemented using the *scikit-learn* [95] and the *DESlib* [96] libraries in Python. For evaluating the results we employ accuracy, sensitivity and specificity for each classifier. The experiment is carried out using 30 replications. For each replication, the datasets are randomly divided as 75% for training and 25% for testing for the original version of the SBDS framework. For the cSBDS framework, the datasets are randomly divided in 50% for the training set, 25% for the validation set and 25% for the test. These divisions are performed preserving the proportion of samples for each class by using a stratified k-fold cross validation. For both the SBDS and the cSBDS, the threshold for the Jaccard coefficient and the containment relationship were set to 90%.

Moreover, for the cSBDS framework, the threshold δ was set initially to 90% to filter the subspaces after they have been merged.

For the DS methods (Table 2.1), the pool of classifiers is composed of 11 decision trees, as suggested by Woloszynski *et al.* (2012) [67], with pruning level set to 10. The pool is generated using the bagging technique, similarly to the methodology followed by Woloszynski in [29, 67].

In the DS methods, following the recent survey on DS techniques [4], the size of the Region of Competence K is set to 7 neighbours for all the techniques based on k-NN. Therefore, to be consistent with this finding for the DS methods, we selected 7 subspaces to predict the unknown test samples in the original SBDS frameworks [42]. Also, because the odd number of classifiers helps to overcome decision ties.

4.4.2 Comparative studies showcasing the details of the changes made to the SBDS framework

In this section, we present the modifications done into the original SBDS framework to create the cSBDS.

4.4.2.1 Comparative study - changes in the k-Nearest Subspace Search (step 2) of the original SBDS version

In this section, we compare the modifications made into the original SBDS framework in regards to how the k-Nearest subspace search is done. The original version of the SBDS framework [42] selected the 7 closest subspaces in relation to each unknown test sample, in terms of the Euclidean Distance and the centroids of the subspaces. In this calculation the authors proposed a multiplying factor that was found empirically to be able to compare the distance within different subspaces with a different number of features. The objective of this study is to answer to the research questions: (1) How the modified SBDS framework performs in high-dimensional small instance dataset? (2) Can the original SBDS framework be improved by changing the k-nearest subspace search step?

The first proposed improvement in this part of the framework was to substitute the distance calculation with a k-NN classifier. One classifier was trained per subspace and evaluated using a nested cross validation with a grid search over the number of nearest neighbours ranging from 7 to 14. The 7 subspaces with the highest accuracy are selected and a majority voting is used to define the label of the unknown test sample. Therefore, by adopting the accuracy of a 7-NN to select the most similar subspaces, we theoretically improved the SBDS framework so it could be applied to different datasets without the need to find empirically to each one the multiplier factor.

Classification accuracy, sensitivity and specificity are reported on Table 4.3. The results in black indicate that there is no statistical difference in terms of the Wilcoxon Rank Sum test, while the results in red indicate that the performance is statistically better and in blue statistically worst with a level of significance of $\alpha = 0.05$. In 50% of the datasets the proposed new method to select the subspaces using a k-NN had a statistically better accuracy, in two cases (Basehock and Relathe) with an improvement over 15%.

The five datasets (Basehock, Colon, Dexter, Leukemia and Relathe), which had a statistical improvement in their performance, are composed of discrete features, whilst the other 5 datasets are composed of continuous features. This could indicate that the multiplier factor employed on the SBDS framework is more sensitive to discrete data.

Table 4.3: Performance metrics (accuracy, sensitivity and specificity) for the original SBDS subspace acquisition and for two subspace selection methods: 1.the distance selection method that was introduced by Maciel-Guerra *et al.* 2020 [42]; 2. the kNN selection modification

		Distance			**kNN**	
	Accuracy	**Sensitivity**	**Specificity**	**Accuracy**	**Sensitivity**	**Specificity**
Allaml	83.70 ± 6.40	52.77 ± 17.78	99.16 ± 2.50	87.67 ± 7.80	62.22 ± 21.05	98.89 ± 3.56
Arcene	79.40 ± 5.35	83.48 ± 6.15	76.19 ± 7.16	79.53 ± 5.05	83.03 ± 7.87	76.79 ± 6.70
Basehock	**75.18 ± 2.43**	**75.75 ± 9.34**	**74.61 ± 9.65**	**91.07 ± 1.69**	**88.47 ± 3.60**	**93.48 ± 3.43**
Colon	**62.92 ± 7.20**	**16.67 ± 13.61**	90.67 ± 9.98	**71.04 ± 8.15**	**43.89 ± 19.95**	87.33 ± 11.23
Dexter	**72.27 ± 3.42**	81.16 ± 11.19	**63.38 ± 12.33**	**80.76 ± 4.46**	84.67 ± 7.17	**76.84 ± 9.18**
Gli85	81.06 ± 7.05	87.78 ± 8.09	66.67 ± 16.22	81.52 ± 6.09	90.44 ± 7.03	62.38 ± 15.86
Leukemia	**83.33 ± 6.42**	**53.89 ± 18.09**	98.06 ± 4.13	**88.52 ± 8.72**	**71.67 ± 25.15**	96.94 ± 5.89
Prostate	82.43 ± 7.34	82.56 ± 9.51	82.31 ± 10.15	82.31 ± 6.99	84.10 ± 7.93	80.51 ± 12.36
Relathe	**66.48 ± 3.36**	86.63 ± 8.31	**49.74 ± 10.12**	**83.86 ± 2.22**	87.80 ± 4.10	**80.58 ± 4.51**
Smkcan	70.92 ± 7.13	80.97 ± 9.48	60.43 ± 12.47	69.22 ± 5.43	78.06 ± 11.33	60.00 ± 12.57

[1] The black colour indicates statistically equivalent, the red colour indicates statistically significant better and the blue colour indicates statistically significant worst. These results are based on the Wilcoxon rank sum test with a confidence level of 95%.

The second proposed improvement in this part of the framework was to substitute the k-NN classifier with a Radial Basis Function Support Vector Machine (RBF-SVM). One classifier was trained per subspace and evaluated using nested cross validation with a grid search for the gamma kernel coefficient $(0.0001, 0.001, 0.01, 0.1)$ and the regularisation parameter $(0.001, 0.01, 0.1, 1, 10, 100, 1000)$.

Classification accuracy, sensitivity and specificity are reported on Table 4.4. The results in black indicate that there is no statistical difference in terms of the Wilcoxon Rank Sum test, while the results in red indicate that the performance is statistically better and in blue statistically worst with a level of significance of $\alpha = 0.05$. In 90% of the datasets the proposed new method to select the subspaces using a RBF-SVM had a statistically better accuracy. Interestingly, the only dataset, which the classification was statistically equal in terms of accuracy, was the Colon dataset. This dataset has discrete features, has the least number of samples (62) and an uneven class distribution (64.5 - 35.5%) which makes it a difficult dataset for any method to learn.

Thus, based on the analysis we can answer the two research questions: the modified version of the SBDS framework does indeed significantly improve the classification performance of the system. In addition, we can see that the use of an RBF-SVM as the classifier method to select the subspace during the second step of the SBDS framework should be considered when compared with the distance metric and the k-NN. This indicates that the use of the empirical factor on the distance metric in the original SBDS framework should be avoided.

4.4.2.2 Comparative study - changes in the subspace clustering selection (step 1) of the original SBDS version

In this section, we compare the modifications made into the original SBDS framework in regards to how the subspace are selected. Given a training dataset, we determine the clusters in each dimension using a GKDE. In the original SBDS version [42], a cluster is defined by a local maximum of the estimated density function, i.e. all points near the local maximum are assigned to the same cluster. Here, the definition of the cluster is

Table 4.4: Performance metrics (accuracy, sensitivity and specificity) for the original SBDS subspace acquisition and for two subspace selection methods: 1 the kNN selection modification; 2. the RBF-SVM selection modification

	kNN			RBF-SVM		
	Accuracy	**Sensitivity**	**Specificity**	**Accuracy**	**Sensitivity**	**Specificity**
Allaml	**87.67 ± 7.80**	**62.22 ± 21.05**	98.89 ± 3.56	**92.78 ± 6.11**	**83.33 ± 15.52**	97.50 ± 4.38
Arcene	**79.53 ± 5.05**	83.03 ± 7.87	**76.79 ± 6.70**	**86.4 ± 4.36**	84.7 ± 7.09	**87.74 ± 4.29**
Basehock	**91.07 ± 1.69**	**88.47 ± 3.60**	**93.48 ± 3.43**	**95.74 ± 0.95**	**94.68 ± 1.62**	**96.80 ± 1.41**
Colon	71.04 ± 8.15	43.89 ± 19.95	87.33 ± 11.23	73.13 ± 11.65	52.22 ± 21.83	85.67 ± 12.57
Dexter	**80.76 ± 4.46**	**84.67 ± 7.17**	**76.84 ± 9.18**	**89.16 ± 2.18**	**90.76 ± 3.25**	**87.56 ± 3.57**
Gli85	**81.52 ± 6.09**	**90.44 ± 7.03**	62.38 ± 15.86	**87.58 ± 5.98**	**96.22 ± 4.77**	69.05 ± 16.94
Leukemia	**88.52 ± 8.72**	**71.67 ± 25.15**	96.94 ± 5.89	**94.63 ± 4.86**	**85.56 ± 14.10**	99.17 ± 2.50
Prostate	**82.31 ± 6.99**	84.10 ± 7.93	**80.51 ± 12.36**	**90.90 ± 5.39**	87.95 ± 7.87	**93.85 ± 5.40**
Relathe	**83.86 ± 2.22**	87.80 ± 4.10	**80.58 ± 4.51**	**88.32 ± 1.41**	87.86 ± 2.66	**88.70 ± 2.72**
Smkcan	**69.22 ± 5.43**	78.06 ± 11.33	60.00 ± 12.57	**71.84 ± 6.26**	81.25 ± 11.06	62.03 ± 12.55

[1] The black colour indicates statistically equivalent, the red colour indicates statistically significant better and the blue colour indicates statistically significant worst. These results are based on the Wilcoxon rank sum test with a confidence level of 95%.

the same, but the search is conducted in each class independently, whilst in the original version all the points were used regardless of their class.

After finding all one-dimensional clusters, a merge process is conducted to obtain the subspace clusters. Here we adopt a general method based on Tian and Gu (2019) [87]. The same method was used in this thesis.

After the merging process, all the one dimensional subspaces are removed and we train one classifier per subspace. Since each subspace is formed by a single class, we decided to use all the samples in the training set and only the features that were selected for the subspace. Next, we tested each classifier on the validation set and selected the ones with an accuracy over a initial threshold δ, if less than 7 subspaces were selected, δ is decreased by 5% until 7 subspaces or more are selected. The objective of this study is to answer to the research questions: (1) How the modified SBDS framework performs in high-dimensional small instance dataset? (2) Can the original SBDS framework be improved by changing subspace clustering step? (3) Can the modified SBDS framework select less features?

Classification accuracy, sensitivity and specificity are reported on Table 4.5 for the original SBDS version [42] against the proposed SBDS framework with a novel way to select the subspaces and with a k-NN as the classifier in the k-Nearest Subspace Search. It is interesting to notice that the three datasets (Basehock, Dexter and Relathe), that achieved a statistically better accuracy performance in the proposed SBDS framework, are text based datasets with discrete features created using a bag-of-words model. These three datasets are also the only ones closer to a 50-50% class distribution. Moreover, in two of these datasets (Dexter and Relathe) the difference between sensitivity and specificity was reduced using the new proposed method, which was the overall aim of creating this new method to select the subspace, i.e. by attempting to find one-dimensional clusters per class we aim to remove the influence of class distribution in the selection of subspaces.

Table 4.5: Performance metrics (accuracy, sensitivity and specificity) for the original SBDS framework and for the modified framework proposed in this work with a novel subspace acquisition method and a kNN subspace selection method

		Original			**Modified**	
	Accuracy	**Sensitivity**	**Specificity**	**Accuracy**	**Sensitivity**	**Specificity**
Allaml	83.70 ± 6.40	52.77 ± 17.78	99.16 ± 2.50	85.56 ± 9.03	59.44 ± 25.34	98.61 ± 3.78
Arcene	79.40 ± 5.35	83.48 ± 6.15	76.19 ± 7.16	78.53 ± 5.80	80.61 ± 8.93	76.90 ± 8.08
Basehock	**75.18 ± 2.43**	75.75 ± 9.34	**74.61 ± 9.65**	**82.57 ± 4.54**	73.68 ± 12.27	**91.49 ± 9.55**
Colon	62.92 ± 7.20	16.67 ± 13.61	90.67 ± 9.98	66.67 ± 9.99	25.56 ± 24.62	91.33 ± 10.24
Dexter	**72.27 ± 3.42**	**81.16 ± 11.19**	**63.38 ± 12.33**	**76.53 ± 4.94**	**74.36 ± 8.92**	**78.71 ± 10.44**
Gli85	81.06 ± 7.05	**87.78 ± 8.09**	66.67 ± 16.22	83.03 ± 7.69	**92.44 ± 7.04**	62.86 ± 17.14
Leukemia	83.33 ± 6.42	53.89 ± 18.09	98.06 ± 4.13	83.33 ± 9.18	56.11 ± 26.35	96.94 ± 6.97
Prostate	82.43 ± 7.34	82.56 ± 9.51	82.31 ± 10.15	82.05 ± 7.84	83.85 ± 8.96	80.26 ± 12.67
Relathe	**66.48 ± 3.36**	**86.63 ± 8.31**	**49.74 ± 10.12**	**78.33 ± 2.99**	**90.21 ± 7.29**	**68.46 ± 4.62**
Smkcan	70.92 ± 7.13	**80.97 ± 9.48**	60.43 ± 12.47	67.94 ± 7.96	**74.58 ± 11.50**	61.01 ± 12.42

[1] The black colour indicates statistically equivalent, the red colour indicates statistically significant better and the blue colour indicates statistically significant worst. These results are based on the Wilcoxon rank sum test with a confidence level of 95%.

Since the proposed modification impacts the number of subspace and consequently the possible number of features selected in each framework we have also analysed the number of subspaces and features in each one. Table 4.6 indicates the number of subspaces and features selected in the original SBDS framework, which is based on a subspace clustering method that uses all the samples, and the modified version, which is based on a subspace clustering method that uses samples from a single class at each time. Although the majority of the datasets (70%) had a statistically equivalent accuracy performance, Table 4.6 shows that in 90% of the datasets the modified version had a lower average of features being selected, with 50% of the datasets having more subspaces. Therefore, based on this analyses we can answer the three research questions proposed for at the beginning of this section: the modified version of the SBDS framework was able to have an statistically equivalent performance on 70% of the datasets and a statistically better performance in 30% of them based on the accuracy level even though it selected less features in 90% of the datasets. This indicates that the modified version is better selecting the most important features.

Table 4.6: Number of subspaces and features comparison between the original SBDS framework and the modified framework proposed in this work with a novel subspace acquisition method

datasets	Subspaces Original	Subspaces Modified	Features	Features Original	Features Modified	Reduction
Allaml	**257.57 ± 7.46**	**146.0 ± 26.04**	7129	**3417.57 ± 65.17**	**2294.63 ± 88.55**	67.81%
Arcene	**325.93 ± 11.75**	**313.47 ± 18.56**	10000	**9884.77 ± 12.87**	**9668.73 ± 27.27**	3.31%
Basehock	**320.8 ± 5.36**	**435.0 ± 9.55**	4862	**4860.97 ± 1.11**	**2823.93 ± 85.32**	41.92%
Colon	**389.5 ± 148.08**	**75.3 ± 22.85**	2000	**1989.5 ± 9.45**	**1703.73 ± 81.89**	14.81%
Dexter	**432.93 ± 5.62**	**495.3 ± 11.14**	20000	**9605.63 ± 68.48**	**3330.23 ± 53.67**	83.35%
Gli	**934.5 ± 16.24**	**359.17 ± 61.1**	22283	**13852.7 ± 197.66**	**10313.77 ± 220.2**	53.71%
Leukemia	**561.57 ± 118.82**	**152.8 ± 22.8**	7070	**7030.57 ± 7.8**	**6945.43 ± 13.08**	1.76%
Prostate	**137.6 ± 12.38**	**164.8 ± 19.06**	5966	**5271.2 ± 74.58**	**5091.43 ± 112.9**	14.66%
Relathe	**336.73 ± 7.26**	**467.03 ± 12.79**	4322	**4321.57 ± 0.56**	**2677.37 ± 59.75**	38.05%
Smkcan	**326.6 ± 19.04**	**2009.0 ± 289.13**	19993	**10382.03 ± 484.32**	**11732.93 ± 541.93**	48.07%

[1] The black colour indicates statistically equivalent, the red colour indicates statistically significant more subspaces/features and the blue colour indicates statistically significant less subspaces/features. These results are based on the Wilcoxon rank sum test with a confidence level of 95% for the comparison of the number of subspaces and the number of features in each framework.
[2] The reduction column indicates the largest reduction in terms of features when compared with the original number of features

4.4.2.3 Comparative study - changes in the k-Nearest Subspace Search (step 2) of the step 1 modified SBDS version

In this section, we aim to evaluate if an RBF-SVM performs better in comparison to a k-NN for the k-Nearest Subspace Search using the modified step 1 SBDS framework discussed in the previous section. The objective of this study is to answer to the research questions: (1) How the modified SBDS framework performs in high-dimensional small instance dataset? (2) Can the modified step 1 SBDS framework be improved by changing the k-nearest subspace search step?

The proposed improvement in this part of the framework was to substitute k-NN classifier with a RBF-SVM. Classification accuracy, sensitivity and specificity are reported on Table 4.7. In 80% of the datasets the use of the RBF-SVM had a statistically better accuracy. Therefore, based on this analyses we can answer the two research questions proposed for at the beginning of this section: the use of the RBF-SVM classifier in the k-Nearest Subspace Search step as shown for the original version of the SBDS framework had a better performance in terms of accuracy when compared with the use of a k-NN.

Table 4.7: Performance metrics (accuracy, sensitivity and specificity) for the proposed modified cSBDS subspace acquisition and for two subspace selection methods: 1. the k-NN selection modification; 2. the RBF-SVM selection modification

	kNN			RBF-SVM		
	Accuracy	**Sensitivity**	**Specificity**	**Accuracy**	**Sensitivity**	**Specificity**
Allaml	**85.56 ± 9.03**	**59.44 ± 25.34**	98.61 ± 3.78	**93.15 ± 5.31**	**83.89 ± 16.38**	97.78 ± 3.69
Arcene	**78.53 ± 5.80**	80.61 ± 8.93	**76.90 ± 8.08**	**85.93 ± 5.33**	83.48 ± 7.56	**87.86 ± 6.94**
Basehock	**82.57 ± 4.54**	**73.68 ± 12.27**	91.49 ± 9.55	**92.04 ± 3.15**	**88.08 ± 4.98**	96.01 ± 3.16
Colon	**66.67 ± 9.99**	**25.56 ± 24.62**	91.33 ± 10.24	**75.00 ± 10.08**	**55.00 ± 23.63**	87.00 ± 11.59
Dexter	**76.53 ± 4.94**	**74.36 ± 8.92**	**78.71 ± 10.44**	**87.60 ± 2.89**	**87.33 ± 6.05**	**87.87 ± 4.47**
Gli85	83.03 ± 7.69	**92.44 ± 7.04**	62.86 ± 17.14	85.76 ± 7.58	**96.22 ± 5.88**	63.33 ± 20.10
Leukemia	**83.33 ± 9.18**	**56.11 ± 26.35**	96.94 ± 6.97	**93.52 ± 6.10**	**80.56 ± 18.30**	100.00 ± 0.00
Prostate	**82.05 ± 7.84**	83.85 ± 8.96	80.26 ± 12.67	**89.87 ± 5.48**	88.21 ± 8.12	**91.54 ± 8.03**
Relathe	**78.33 ± 2.99**	**90.21 ± 7.29**	**68.46 ± 4.62**	**83.52 ± 2.55**	**88.15 ± 3.04**	**79.68 ± 3.48**
Smkcan	67.94 ± 7.96	74.58 ± 11.50	61.01 ± 12.42	65.67 ± 7.82	74.17 ± 9.77	56.81 ± 13.70

[1] The black colour indicates statistically equivalent, the red colour indicates statistically significant better and the blue colour indicates statistically significant worst. These results are based on the Wilcoxon rank sum test with a confidence level of 95%.

4.4.2.4 Comparison with the state-of-art DS techniques and majority voting

In this section, we compared the accuracy obtained by the proposed cSBDS against eleven state-of-art dynamic selection techniques presented previously in Table 2.1 and the majority voting technique. The objective of this study is to answer to the research questions: (1) How the cSBDS framework performs in high-dimensional small instance dataset? (2) How the performance of the cSBDS framework compare with traditional DS methods and majority voting?

The DS methods used in this analysis are: Classifier Rank (CR) [58], Overall Local Accuracy (OLA) [59], Local Class Accuracy (LCA) [59], Multiple Classifier Behaviour (MCB) [61], Modified Local Accuracy (MLA) [62], K-Nearest Oracles - Eliminate (KNORA-E) [7], K-Nearest Oracles - Union (KNORA-U) [7], k-Nearest Output Profiles - Elimiante (KNOP-E) [68], k-Nearest Output Profiles - Union (KNOP-U) [68], Meta-Learning - DES (Meta-DES) [30].

The Friedman test with Iman-Davenport correction (F_F) [97] is employed for statistical comparison of multiple classifier system techniques as suggested by Cruz and Demsar in [4, 98]. The null-hypothesis states that all algorithms are equivalent and so their average ranks should be equal.

The rank of each method is calculated using the average ranking approach. The best performing algorithm is the one with the lowest average rank. Next, as suggested by [98], to compare all classifiers against a control, we use the z value (Equation 4.4 to compare two classifiers. The z value is used to find the corresponding probability from the table of normal distribution, which is then compared with an appropriate α.

$$z = (R_i - R_j) \Bigg/ \sqrt{\frac{k(k+1)}{6N}} \tag{4.4}$$

where R_i is the rank of i-th classifier, k is the number of classifiers and N is the number of datasets. The z value is than used to find the corresponding p-value from the two-tailed normal distribution table, which is subsequently compared to an appropriate significance level α.

With 13 classifiers and 10 datasets, the Friedman test is distributed according to the F distribution with $13 - 1 = 12$ and $(10 - 1) \times (13 - 1) = 108$ degrees of freedom. The critical value of F(12,108) for $\alpha = 0.0001$ is 3.7324.

The results of the mean accuracy and the rank of each classifier over each dataset is shown on Table 4.8. The F_F statistic found was 27.6104, so the null-hypothesis can be rejected with 99.99% confidence. To compare all classifiers against a control, in this case the majority voting approach we measure the Z-score for each classifier. The cS-BDS method was the only one that achieved a statistically significant lower rank when compared to the majority voting with 95% confidence.

Therefore to answer to the research question, these results showcases the ability of the SBDS framework to uncover the most important features in a small instance high-dimensional dataset. Maciel-Guerra *et al.* (2020) [9] showed that DS methods fail to perform on this type of datasets, because the k-NN approach, commonly adopted to define regions of competence, deteriorates as the number of dimensions increases. Therefore, the use of a subspace clustering method as the "region of competence" in the DS method enabled the cSBDS framework to achieve a statistically significant results when compared with majority voting.

Table 4.8: Mean and Rank results of the accuracy obtained by the proposed mSBDS, 11 state-of-art DS methods and Majority Voting

	CR	OLA	LCA	MCB	MLA	KNORA-E	KNORA-U	KNOP-E	KNOP-U	Meta-DES	DES-P	mSBDS	Majority Voting
Arcene	66.40% (11)	67.13% (10)	68.27% (7.5)	67.60% (9)	61.20% (13)	68.27% (7.5)	71.73% (2)	66.33% (12)	70.67% (4)	69.47% (6)	70.40% (5)	85.93% (1)	71.00% (3)
Allaml	84.81% (12)	88.89% (8)	87.41% (11)	87.59% (10)	84.44% (13)	91.30% (6)	92.96% (3.5)	89.07% (7)	93.33% (1)	88.70% (9)	92.96% (3.5)	93.15% (2)	92.78% (5)
Basehock	89.49% (5)	89.30% (7)	88.88% (12)	89.23% (9)	88.48% (13)	89.46% (6)	89.22% (10)	91.28% (2)	90.45% (3)	89.65% (4)	89.23% (8)	92.04% (1)	89.17% (11)
Colon	73.13% (7.5)	73.13% (7.5)	71.46% (11)	70.63% (12)	70.42% (13)	73.96% (6)	76.04% (1)	72.29% (10)	75.42% (2)	72.92% (9)	74.17% (5)	75.00% (3)	74.38% (4)
Dexter	78.62% (13)	81.67% (9)	80.29% (11)	81.02% (10)	79.40% (12)	84.07% (7)	87.22% (5)	84.27% (6)	87.89% (1)	83.24% (8)	87.31% (3)	87.60% (2)	87.29% (4)
Gli	74.55% (12)	76.06% (9)	74.55% (11)	75.00% (10)	73.64% (13)	76.82% (7.5)	82.12% (3)	76.82% (7.5)	81.36% (5)	77.73% (6)	82.73% (2)	85.76% (1)	81.52% (4)
Leukemia	80.19% (13)	88.33% (8)	87.41% (9)	85.74% (11)	82.04% (12)	88.52% (7)	90.93% (2)	90.37% (5)	90.56% (3.5)	87.04% (10)	90.56% (3.5)	93.52% (1)	90.00% (6)
Prostate	79.74% (12)	83.85% (9)	80.00% (11)	82.95% (10)	77.44% (13)	84.87% (6)	88.33% (2)	84.10% (7)	87.82% (5)	83.97% (8)	88.21% (3.5)	89.87% (1)	88.21% (3.5)
Relathe	78.94% (10)	79.40% (9)	78.43% (12)	78.69% (11)	77.50% (13)	80.94% (5)	80.85% (6)	81.21% (3.5)	82.50% (2)	80.04% (8)	81.21% (3.5)	83.52% (1)	80.76% (7)
Smkcan	58.79% (9)	58.58% (11)	60.71% (7)	59.22% (8)	58.58% (11)	58.44% (13)	62.55% (4)	58.58% (11)	62.98% (2)	60.92% (6)	62.62% (3)	65.67% (1)	61.35% (5)
Average Rank	10.45	8.75	10.25	10	12.6	7.1	3.85	7.1	2.85	7.4	4	1.4	5.25
p-value	**0.00283**	**0.04447**	**0.00409**	**0.00639**	**0.00002**	0.28814	0.42149	0.28814	0.1682	0.21703	0.47294	**0.02707**	1

[1] The numbers in brackets indicates the rank of the classifier for each dataset
[2] The test statistics for comparing the i-th and j-th classifier is shown in Equation 4.4 with their pvalues shown here.
[3] The values highlighted in red are statistically significant with a 95% confidence.

4.4.3 Conclusions

In this section, we presented the modified version of the SBDS framework called cSBDS. It first searches for one-dimensional clusters using a GKDE and then a merging procedure is conducted to generate subspace clusters. A filter approach is used based on the performance of a classifier to select the best subspaces. Next, the selected classifiers are trained on each subspace. Finally, the performance of the best classifiers in relation to the unknown test sample on each subspace is used to determine which subspaces will be used to make the prediction.

Experiments were conducted using 10 small instance high-dimensional problems. First, we performed an analysis modifying the k-Nearest Subspace Search (step 2) of the original SBDS framework proposed by Maciel-Guerra *et al.* (2020) [42] and presented on Section 4.3. The analysis demonstrated that the empirical factor and the distance approach proposed in the original version could be substituted by a RBF-SVM classifier which achieved a statistically better accuracy in 100% of the datasets when compared to the original version and 90% when compared to the k-NN classifier. Second, we performed an analysis modifying the subspace clustering selection (step 1) of the original SBDS framework [42]. The analysis demonstrated that by changing how the subspaces are selected we could decrease the number of feature being selected in 90% of the datasets and increase the number of subspaces in 50% of the datasets without loosing performance and in three datasets having a statistically better performance in terms of accuracy. Third, the changed the classifier in modified step 1 SBDS framework to a RBF-SVM (cSBDS) and compared to the k-NN approach. In this analysis, the cSBDS framework achieved a statistically better accuracy in 80% of the datasets, indicating once again that the use of a RBF-SVM can statistically increase the performance of the SBDS framework. Finally, the performance obtained by the cSBDS framework was compared with 11 state-of-art DS techniques and with the majority voting. Experimental results demonstrate that the cSBDS outperforms the studied DS techniques and majority voting. In addition, the gain in performance obtained by the cSBDS framework is shown to be statistically significant when compared to the majority voting based on the Friedman test with a post-hoc z-score test.

This results confirm the hypothesis that the cSBDS is able to select the appropriate features and have high performances in small instance high-dimensional datasets when compared to state-of-art DS techniques and the majority voting.

The performance of cSBDS has been demonstrated on real world datasets in this chapter. On Chapter 5, we use synthetic datasets to better understand why cSBDS performs the way it does and if its able to correct find the most important features using the subspace clustering feature selection.

4.5 Chapter Summary

In this chapter, the Subspace-Based Dynamic Selection (SBDS) framework has been presented. In particular, each component of the theoretical framework has been discussed and two versions are presented with differences on how the subspaces are found and selected. The proposed method merges the capabilities of the DS framework with a subspace clustering approach to improve feature selection by finding the most important features for each subspace in high-dimensional small-instances datasets while maintaining or increasing the performance when compared with DS methods. The first proposed SBDS framework had some limitations in relations of how the subspaces were selected and specially the use of an empirical multiplier factor to remove a bias of the distance calculation to find the closest subspaces to the unknown test samples. On the other hand, the proposed cSBDS framework overcomes these limitations and is able to perform statistically better when compared to DS methods and majority voting. Moreover, by changing how the subspaces are found, the cSBDS framework is able to select less features and more subspaces. The next chapter demonstrates how effective this framework can be on synthetic data to find the most important features.

Chapter 5

Subspace Feature Selection Analysis

***

5.1 Introduction

In this chapter, we analyse the features selected by the cSBDS framework introduced in Chapter 4, which aims to evaluate if the subspace clustering method in the SBDS framework can be used as a feature selection approach. To achieve this, we propose to evaluate the cSBDS framework in multiple synthetic datasets varying the classification difficulty and the number of clusters per class in each one; whilst comparing its performance with an RBF-SVM.

The core aim of this thesis is to provide a framework to evaluate high-dimensional small instance datasets using a DS framework with the addition of a subspace clustering method to improve the selection of the features. Therefore, this chapter evaluates if the subspace clustering approach proposed on the cSBDS framework selects the most important features for a given dataset. By confirming this aspect of the cSBDS framework we can acknowledge that it helps with knowledge discovery which in biological datasets, genomics, for instance, can help understand which genes are more important for each sample. This chapter aims to answer one of our main research questions: **"How can we extract information in terms of feature importance for classification for each unseen sample?"**

In this section, we analyse the features selected by the subspace clustering approach on the cSBDS framework introduced in Chapter 4. To achieve this we create multiple synthetic classification datasets with different classification difficulties and a different number of clusters per class. Also, a new approach to finding the subspace clusters is proposed.

This chapter first analyses the feature selection aspect of the subspace clustering method on the cSBDS framework using different synthetic datasets. Next, it compares the features selected by the cSBDS on the proposed synthetic datasets with the ones selected by feature importance methods such as permutation, Gini and statistical importance.

5.2 Overview

High-dimensional feature spaces make a k-NN fail to work [8, 9, 108]. Moreover, since the intrinsic dimensionality is typically small on a high-dimensional dataset, feature selection approaches aim to find a low dimensional feature space that preserves the intrinsic data structure by removing the noisy, irrelevant and redundant features [108].

To choose a good feature subset, many features selection algorithms were proposed: filter, wrapper, embedded and hybrid. The wrapper methods are computationally expensive and can overfit on small training sets. The filter methods are usually a good option for high-dimensional datasets [109]. Nonetheless, according to Tian and Gu (2019) [87] some features might only work for a subset of samples and appear as noise for the rest of the samples, and this phenomenon is more common in high-dimensional datasets.

In 2001, Pechenizkiy *et al.* [38] created the FEDIC (Feature Extraction for Dynamic Integration of Classifiers) algorithm. It is based on early dynamic integration methods while using a PCA and two eigenvector-based class-conditional for feature extraction. Because FEDIC is based on feature extraction methods, it loses the physical information of the original feature set which makes it difficult to understand which of the original features were more important for the classification. Moreover, this method was tested only in datasets that have more samples than features. Nonetheless, Pechenizkiy *et al.* (2001) [38] were able to show that dimensionality reduction methods incorporated into the DS framework can improve the performance of DS methods on some datasets and overcome some of the problems related to high-dimensional datasets.

The SBDS and the cSBDS frameworks proposed in Chapter 4 use a subspace clustering approach to select which features will be used for the classification. Subspace clustering is a technique that finds clusters within different subspaces of one or more dimensions for a specific set of samples [10]. To evaluate if the proposed subspace clustering is selecting the most important features, we first evaluated the performance of an RBF-SVM with and without a filter approach and compared it with the cSBDS performance on synthetic datasets. Next, we compare the features selected by the cSBDS framework with other feature importance methods. Finally, we compare the performance of two cSBDS methods on real-world datasets.

5.3 Methodology

To analyse if the cSBDS is selecting the most important features, we decided to create multiple random 2-class classification problems. To achieve this, we used the *make_classification* function from the Scikit-learn library [95] which was adapted from Guyon (2003) [110] to generate the "Madaleo" dataset. The dataset is formed by creating clusters of points normally distributed ($std = 1$) on vertices of an n-informative-dimensional hyper-cube with sides of length $2 * class\_sep$ and assigns an equal c number of clusters to each class. We have created multiple datasets with 200 samples, 400 features (5 of them being informative and 395 useless features drawn at random). The *class_sep* parameter is the class separation which is the multiplying factor of the hyper-cube size, i.e. larger values spread out the clusters/classes and make the classification easier. We have set this parameter to range from 0.1 to 3 with 0.1 steps. Moreover, we have created datasets with the number of clusters per class ranging from 1 to 10.

To evaluate the results, we measured the accuracy, sensitivity and specificity of each classifier. The experiment is carried out using 30 replications. The datasets are randomly divided in 50% for the training set, 25% for the validation set and 25% for the test. To preserve the proportion of samples for each class a stratified k-fold cross validation is used. For the cSBDS, the thresholds for the Jaccard coefficient and the containment relationship are set to 90%. Moreover, for the cSBDS framework, the threshold δ is set initially to 90% to filter the subspaces after they have been merged.

For the RBF-SVM classifiers, the following set of values are employed for the hyper-parameters, before and after feature selection: γ (RBF kernel coefficient) = [0.0001, 0.001, 0.01, 0.1] and C (L2 penalty parameter) = [0.001, 0.01, 0.1, 1, 10, 100, 1000]. The feature selection approach used for the RBF-SVM classifier is the filter approach with a Wilcoxon rank-sum test which tests the null hypothesis that two sets of measurements are drawn from the same distribution.

Concerning the comparison of feature importance metrics, the following methods are employed: (1) Gini importance using a Random Forest classifier with 10 estimators and a maximum depth of the trees of 10; (2) permutation importance using as a base model the previous random forest classifier with 5 repeats; (3) Wilcoxon rank-sum test; (4) chi-square test. The first two methods are estimated using a stratified k-fold cross validation with 75% for the training set and 25% for the test set. The last two methods are evaluated using all the samples.

5.4 Results and Discussion

Feature selection uncovers the most important features that can provide insights into the nature of the studied problem. The objective of studying feature selection is to remove unwanted, irrelevant and redundant features that contribute neither to the prediction of a target class nor to the efficiency of data mining methods. Unlike other algorithms that use statistical methods to identify the most important features, the subspace clustering approach proposed for the SBDS and cSBDS dynamically selects the clusters, and consequently the most important features, using a Gaussian kernel density estimator method.

In this section, we propose to use multiple synthetic datasets. The performance of the cSBDS framework is compared with the performance of the RBF-SVM classifier with and without a filter feature selection approach. The RBF-SVM classifier was chosen since is the same classifier used to make the predictions in the cSBDS framework, therefore by applying the same data splits into both methods, the only difference lies in how the features are selected. Hence, an improvement in the performance should indicate whether the subspace clustering method is selecting the most important features. To test this hypothesis, the datasets were also evaluated using feature importance methods and their results were compared with the cSBDS approach.

5.4.1 Performance comparison

Figures 5.1, 5.3 and 5.5 show the performance in terms of accuracy, sensitivity and specificity, respectively, for the cSBDS framework with a Gaussian kernel density estimator (cSBDS GKDE), the RBF-SVM using all the features (RBF-SVM) and the RBF-SVM using a filter feature selection approach based on the Wilcoxon rank sum test (RBF-SVM FS). In addition, to compare the performance results between the different methods, a Wilcoxon rank sum test was used and the p-values results are indicated on Figures 5.2, 5.4 and 5.6 for the accuracy, sensitivity and specificity, respectively.

Figures 5.1 and 5.2, first indicate that the RBF-SVM FS when compared with the RBF-SVM performs statistically better in 38.67% of the datasets, while the remaining 61.13% are statistically equivalent. This indicates that the filter feature selection approach was able to improve the performance of the RBF-SVM in terms of accuracy. Next, when comparing the performance of the cSBDS GKDE framework with the RBF-SVM, the first performs better in 26.67% of the datasets, while the second performs better in 4.33% and on 69% they are statistically equivalent. Interestingly, the RBF-SVM performs better only when there is 1 cluster per class, by increasing the number of clusters per class the cSBDS GKDE performs better. Nonetheless, when comparing the cSBDS GKDE with the RBF-SVM FS approach the first performs better in 2% of the datasets, while the second performs better in 14% and on 84% they are statistically equivalent.

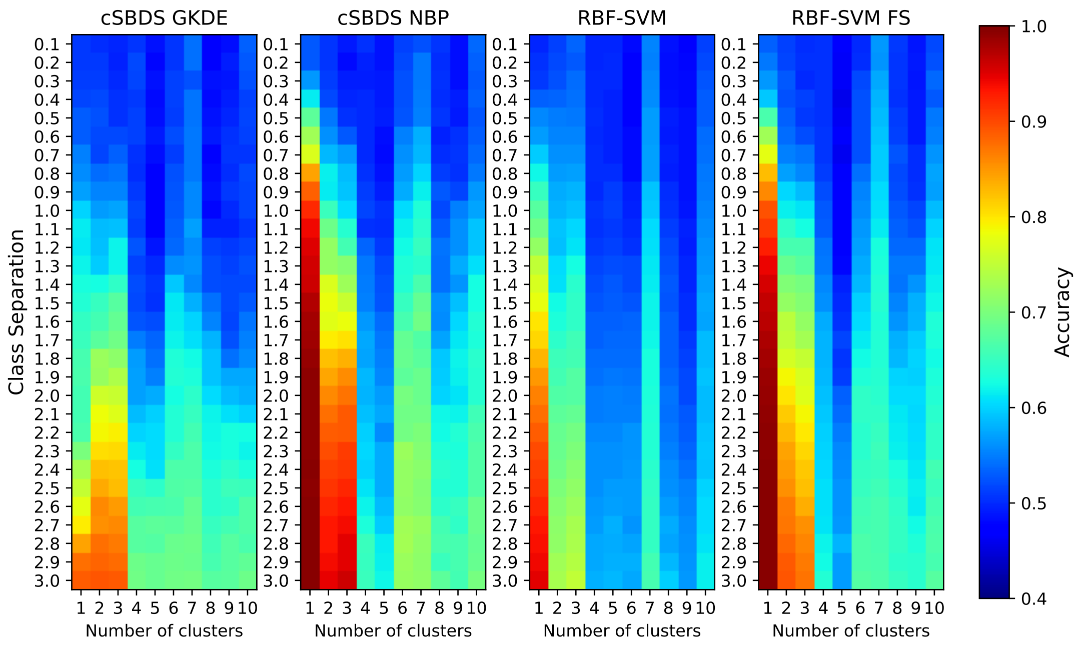


Figure 5.1: Performance results in terms of accuracy for the RBF-SVM classifier using all the features (RBF-SVM), the RBF-SVM classifier with filter feature selection approach (RBF-SVM FS) and the cSBDS framework with a Gaussian kernel density estimator (cSBDS GKDE) and a Jenks natural break points (cSBDS NBP) to determine the one-dimensional clusters in the subspace clustering selection. Each method is evaluated on dataset with different classification difficulties (class separation ranging from 0.1 - 3) and different numbers of clusters per class (ranging from 1 - 10).

Figures 5.3 and 5.4 indicate the performance in terms of the sensitivity. They show that the RBF-SVM FS when compared with the RBF-SVM performs statistically better in 32% of the datasets, while the RBF-SVM performs better on 1.33% and the remaining 66.67% are statistically equivalent. This indicates that the filter feature selection approach was able to improve the performance of the RBF-SVM in terms of sensitivity. Next, when comparing the performance of the cSBDS GKDE framework with the RBF-SVM, the first performs better in 30.67% of the datasets, while the second performs better in 3.33% and on 66% they are statistically equivalent. The RBF-SVM performs better in terms of sensitivity, when the number of clusters are increased. When comparing the cSBDS GKDE with the RBF-SVM FS approach the first performs better in 3.67% of the

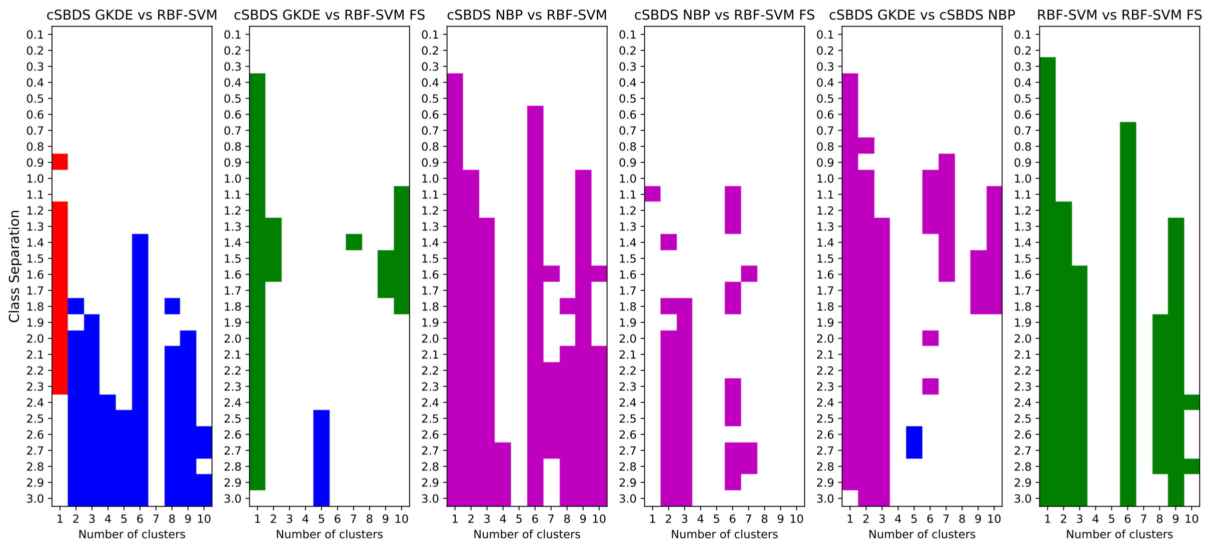


Figure 5.2: Statistical comparison of the accuracy performance of all the combinations of the methods RBF-SVM, RBF-SVM FS (RBF-SVM with a filter feature selection), cSBDS GKDE and cSBDS NPB using a Wilcoxon rank sum test. The colour white indicates that the null hypothesis was not rejected with a 99.999% confidence. The colours red (RBF-SVM), green (RBF-SVM FS), blue (cSBDS GKDE) and magenta (cSBDS NBP) indicate that method performed statistically better with a 99.999% confidence.

datasets, while the second performs better in 9.67% and on 86.67% they are statistically equivalent.

Moreover, the Figures 5.5 and 5.6 indicate the performance in terms of the specificity. They show that the RBF-SVM FS when compared with the RBF-SVM performs statistically better in 17.33% of the datasets, while the remaining 82.67% are statistically equivalent. This indicates that the filter feature selection approach was able to improve the performance of the RBF-SVM in terms of specificity. Next, when comparing the performance of the cSBDS GKDE framework with the RBF-SVM, the first performs better in 7.67% of the datasets, while the second performs better in 14.33% and on 78% they are statistically equivalent. The RBF-SVM performs better in terms of specificity, when the number of cluster per class increase and on more difficult classification datasets (low class separation). Nonetheless, when comparing the cSBDS GKDE with the RBF-SVM FS approach the RBF-SVM FS performs better in 8.67% of the datasets, while the remaining 91.33% are statistically equivalent.

The analysis of the results indicate that the performance increases when the classification difficulty decreases (higher class separation), as we would except. Also, by

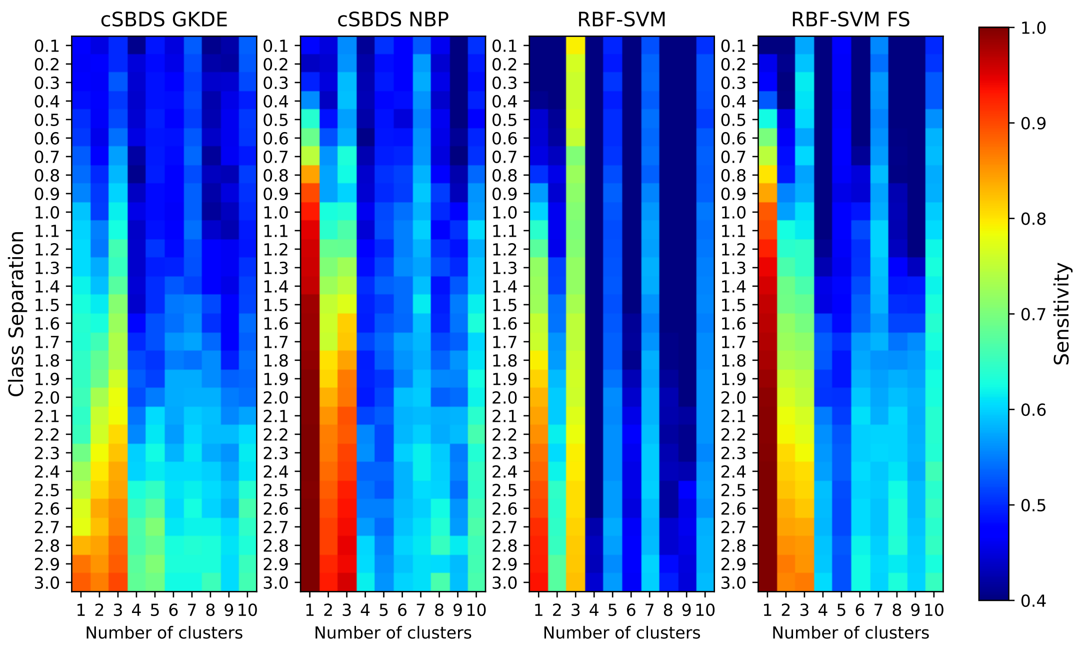


Figure 5.3: Performance results in terms of sensitivity for the RBF-SVM classifier using all the features (RBF-SVM), the RBF-SVM classifier with filter feature selection approach (RBF-SVM FS) and the cSBDS framework with a Gaussian kernel density estimator (cSBDS GKDE) and a Jenks natural break points (cSBDS NBP) to determine the one-dimensional clusters in the subspace clustering selection. Each method is evaluated on dataset with different classification difficulties (class separation ranging from 0.1 - 3) and different numbers of clusters per class (ranging from 1 - 10).

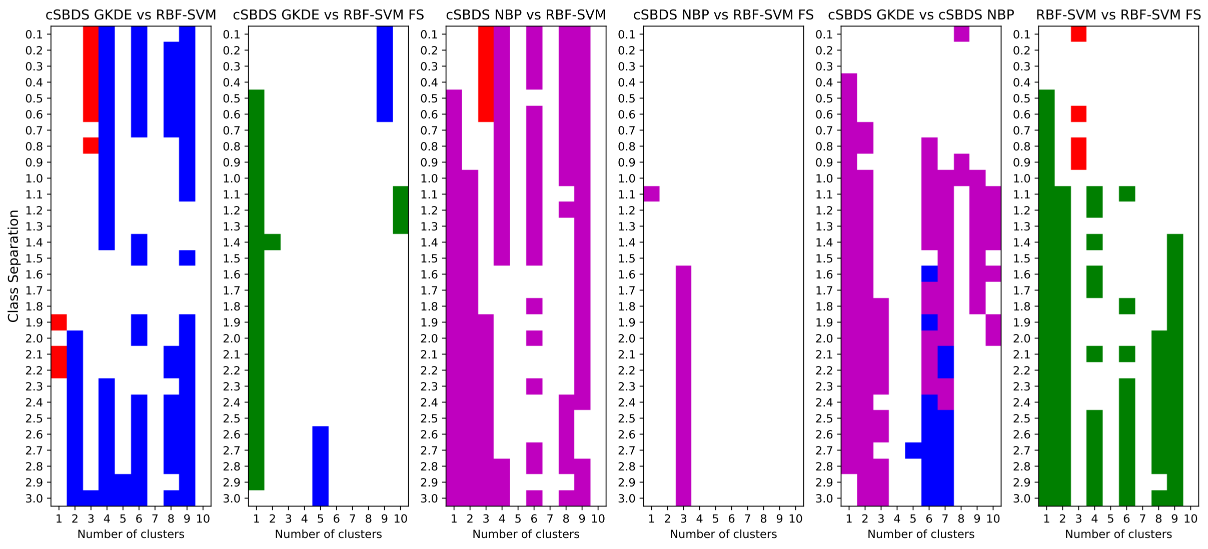


Figure 5.4: Statistical comparison of the sensitivity performance of all the combinations of the methods RBF-SVM, RBF-SVM FS (RBF-SVM with a filter feature selection), cSBDS GKDE and cSBDS NPB using a Wilcoxon rank sum test. The colour white indicates that the null hypothesis was not rejected with a 99.999% confidence. The colours red (RBF-SVM), green (RBF-SVM FS), blue (cSBDS GKDE) and magenta (cSBDS NBP) indicate that method performed statistically better with a 99.999% confidence.

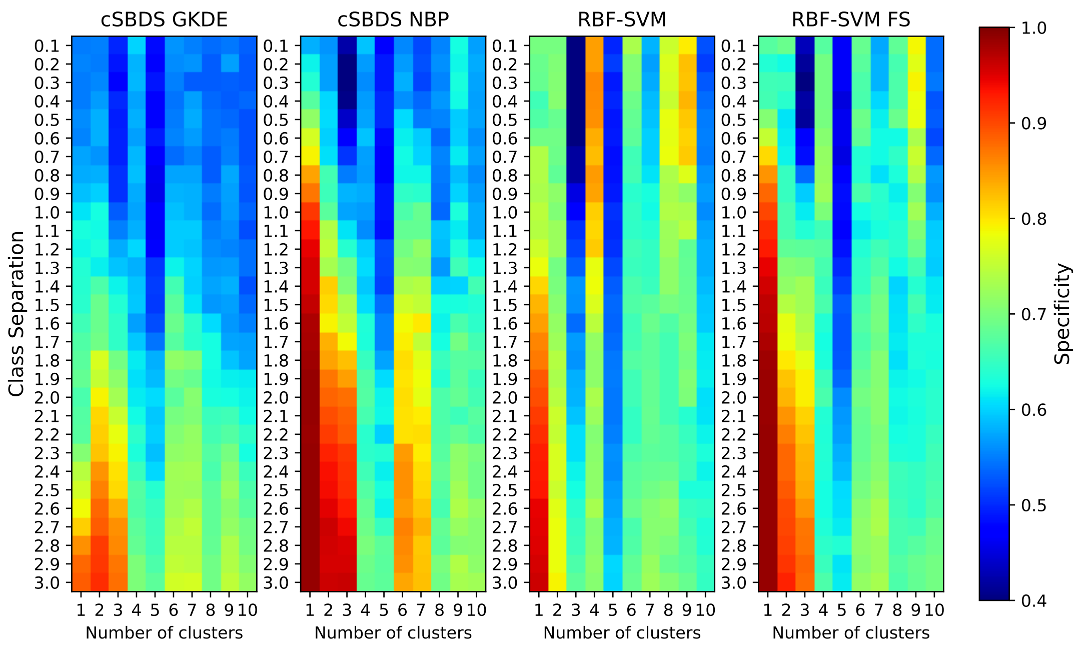


Figure 5.5: Performance results in terms of specificity for the RBF-SVM classifier using all the features (RBF-SVM), the RBF-SVM classifier with filter feature selection approach (RBF-SVM FS) and the cSBDS framework with a Gaussian kernel density estimator (cSBDS GKDE) and a Jenks natural break points (cSBDS NBP) to determine the one-dimensional clusters in the subspace clustering selection. Each method is evaluated on dataset with different classification difficulties (class separation ranging from 0.1 - 3) and different numbers of clusters per class (ranging from 1 - 10).

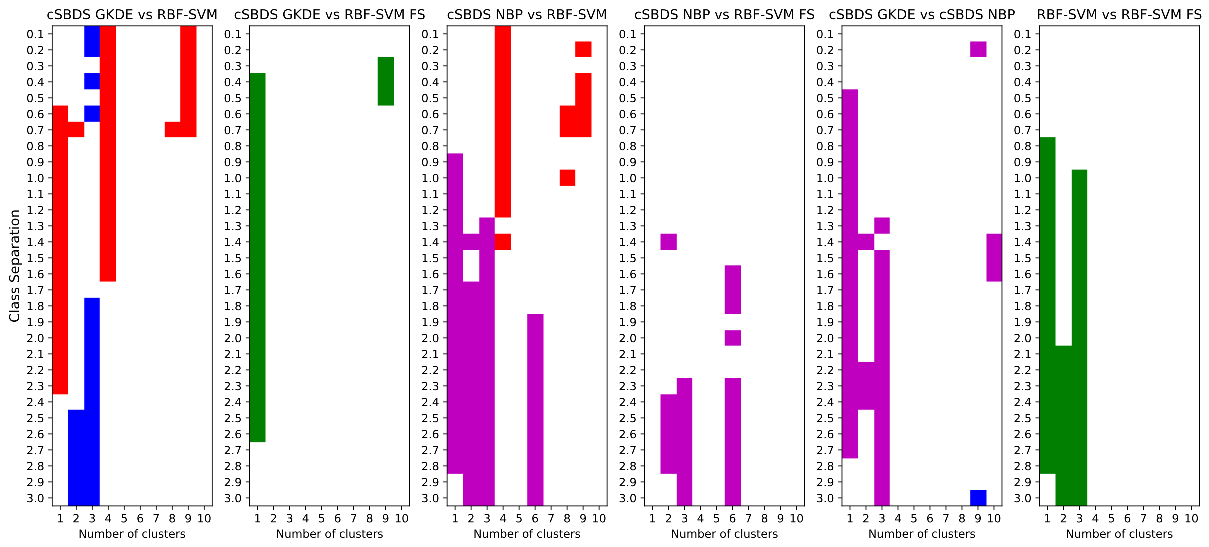


Figure 5.6: Statistical comparison of the specificity performance of all the combinations of the methods RBF-SVM, RBF-SVM FS (RBF-SVM with a filter feature selection), cSBDS GKDE and cSBDS NPB using a Wilcoxon rank sum test. The colour white indicates that the null hypothesis was not rejected with a 99.999% confidence. The colours red (RBF-SVM), green (RBF-SVM FS), blue (cSBDS GKDE) and magenta (cSBDS NBP) indicate that method performed statistically better with a 99.999% confidence.

increasing the number of clusters per class the performance decreases for the same classification difficulty for all the methods. In addition, the Figures 5.1 and 5.6 indicate that for the majority of the datasets the cSBDS GKDE performs statistically equivalent to the RBF-SVM FS. Also, for the datasets that have a statistical different performance, the RBF-SVM FS tends to perform better than the cSBDS GKDE. Therefore, the RBF-SVM FS would be the preferred choice since it is selecting the most important features for the classification in a less complex way due to the nature of the filter approach regardless of which performance metric is used. We hypothesise that the issue of the cSBDS GKDE could be on how the one-dimensional cluster is select by using a Gaussian kernel density estimator. To test this hypothesis, we substitute the GKDE approach with the Jenks Natural Break Points (NBP) method.

The Jenks natural break points classification method is a clustering method used to determine the best arrangement of values into different classes. This is achieved by minimising each class's average deviation from the mean, while maximising each class's deviation from the means of the other classes, i.e. it reduces the variance within classes and increases the variance between classes [111].

Figure 5.7 shows an example of the difference between the GKDE and the NBP methods. In this example, the GKDE would separate the data into two different one-dimensional clusters, while the NBP would create three different one-dimensional clusters: one cluster for the class 1 (red) and 2 clusters for class 2 (black). The NBP tends to split the data into smaller and more diverse clusters which could separate better the samples into different subspaces.

Figures 5.1 and 5.2 show the results in terms of accuracy and they indicate that the cSBDS with a NBP method performed statistically better in 48.33% of the datasets against the RBF-SVM, in 13.67% against the RBF-SVM FS and 30.67% against the cSBDS GKDE. In addition it performed statistically equivalent in 51.67% of the datasets against the RBF-SVM, in 86.33% against the RBF-SVM FS and 68.67% against the cSBDS GKDE. Only in 0.67% of the datasets against the cSBDS GKDE, the cSBDS NBP had a statistically lower performance.

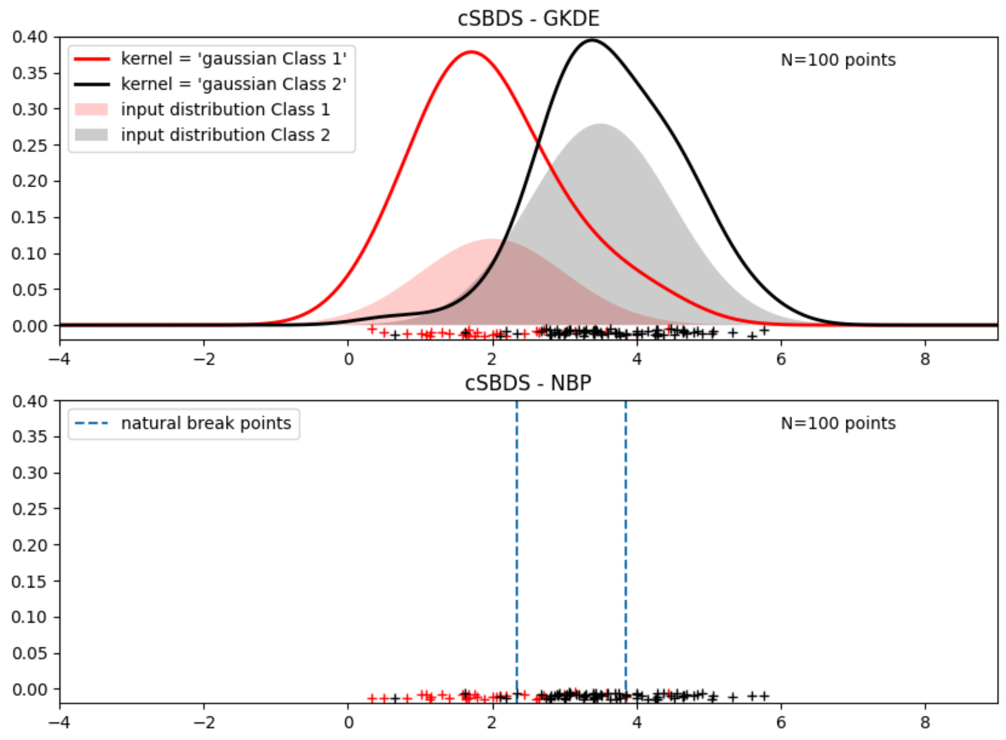


Figure 5.7: Comparison between the Gaussian kernel density estimator (GKDE) and the Jenks natural break points (NBP) methods for the one-dimensional clustering definition in the cSBDS framework.

Figures 5.3 and 5.4 show the results in terms of sensitivity and they indicate that the cSBDS NBP performed statistically better in 47% of the datasets against the RBF-SVM, in 5.33% against the RBF-SVM FS and 34.67% against the cSBDS GKDE. In addition it performed statistically equivalent in 51% of the datasets against the RBF-SVM, in 94.67% against the RBF-SVM FS and 59.33% against the cSBDS GKDE. Only in 2% of the datasets against the RBF-SVM and in 6% against the cSBDS GKDE, the cSBDS NBP had a statistically lower performance.

Figures 5.5 and 5.6 show the results in terms of specificity and they indicate that the cSBDS NBP performed statistically better in 21.67% of the datasets against the RBF-SVM, in 8.67% against the RBF-SVM FS and 16% against the cSBDS GKDE. In addition it performed statistically equivalent in 71.33% of the datasets against the RBF-SVM, in 91.33% against the RBF-SVM FS and 83.67% against the cSBDS GKDE. Only in 7% of the datasets against the RBF-SVM and in 0.33% against the cSBDS GKDE, the cSBDS

NBP had a statistically lower performance.

These results indicate that in terms of accuracy, sensitivity and specificity, the substitution of the GKDE method for the NBP resulted in a statistical better framework for this type of dataset. This confirms our hypothesis that the Gaussian kernel density estimator had a bias when used on a Gaussian based dataset, and that the use of a different method, like the Jenks Natural Break Points (NBP), improved the classification performance.

5.4.2 Feature importance methods comparison

To evaluate the benefits of using the cSBDS framework as a feature selection approach, we compared the features selected by it with traditional feature importance methods such as permutation, Gini and statistical importance. Feature importance methods refer to techniques that calculate a score for all the features for a given method. These scores simply represent the importance of each feature. Normally, a higher score means that the specific feature will have a larger effect on the model that is being used. On statistical methods, the p-value given by them is used to indicate the importance of the features, therefore the lower the p-value is the more important the feature becomes.

Feature importance methods are important to use for several reasons: (1) they help understand better the data that is used as an input for the model, showing which are the relevant features; (2) by selecting the features through their importance, the number of features is reduced making the model simpler but also speeding up the prediction and ultimately improving the performance of the model; and (3) by having the importance of the features it is easier to interpret and communicate to other researchers which features have the most predictive power to the model.

The Gini importance is used to calculate the node impurity and the importance is measured as a reduction in the impurity weighted by the number of samples that reach that node. This is also known as the node probability [112]. In addition, the permutation importance is calculated by noticing the increase or decrease in error when the values of a feature are permuted. If permuting the values of a feature causes a huge change in the error, it means that this feature is important for the model [112].

Figures 5.8, 5.9 and D.1-D.8 show the importance of the 400 features on each of the synthetically created datasets used in the previous analysis. Each figure corresponds to the datasets created with a specific number of clusters per class (ranging from 1 to 10), while the panels inside each figure show the importance of the 400 features on each dataset varying the class separation parameter from 0.1 to 3. with 0.1 steps. In each figure, the results of 6 importance methods are shown: (1) p-value of the Wilcoxon ranksum test, (2) p-value of the chi-square test, (3) permutation importance, (4) Gini importance, (5) cSBDS with a GKDE and (6) cSBDS with a NBP. Each dataset contains five features that are important, while the other 395 features are considered useless. Also, they were created to maintain the same important features while varying the class separation parameter.

The results on Figures 5.8, 5.9 and D.1-D.8 indicate that (1) the importance of the features increase in all methods when the class separation increases, as expected; and (2) when changing the number of clusters per class the Gini, permutation and statistical importance methods reduce their capability of differentiating between the informative and the useless features, while the cSBDS NBP method still indicates the most important features regardless of the number of clusters per classes in the data. Interestingly, when comparing the importance of the features calculated by the cSBDS GKDE and the cSBDS NBP methods, the last one has large difference between the features that are not important (close to 0) and the ones that are important (close to 100). In both cases, the importance is given by the selection frequency, i.e. the percentage of times a feature is selected to give the final prediction of the unseen test samples.

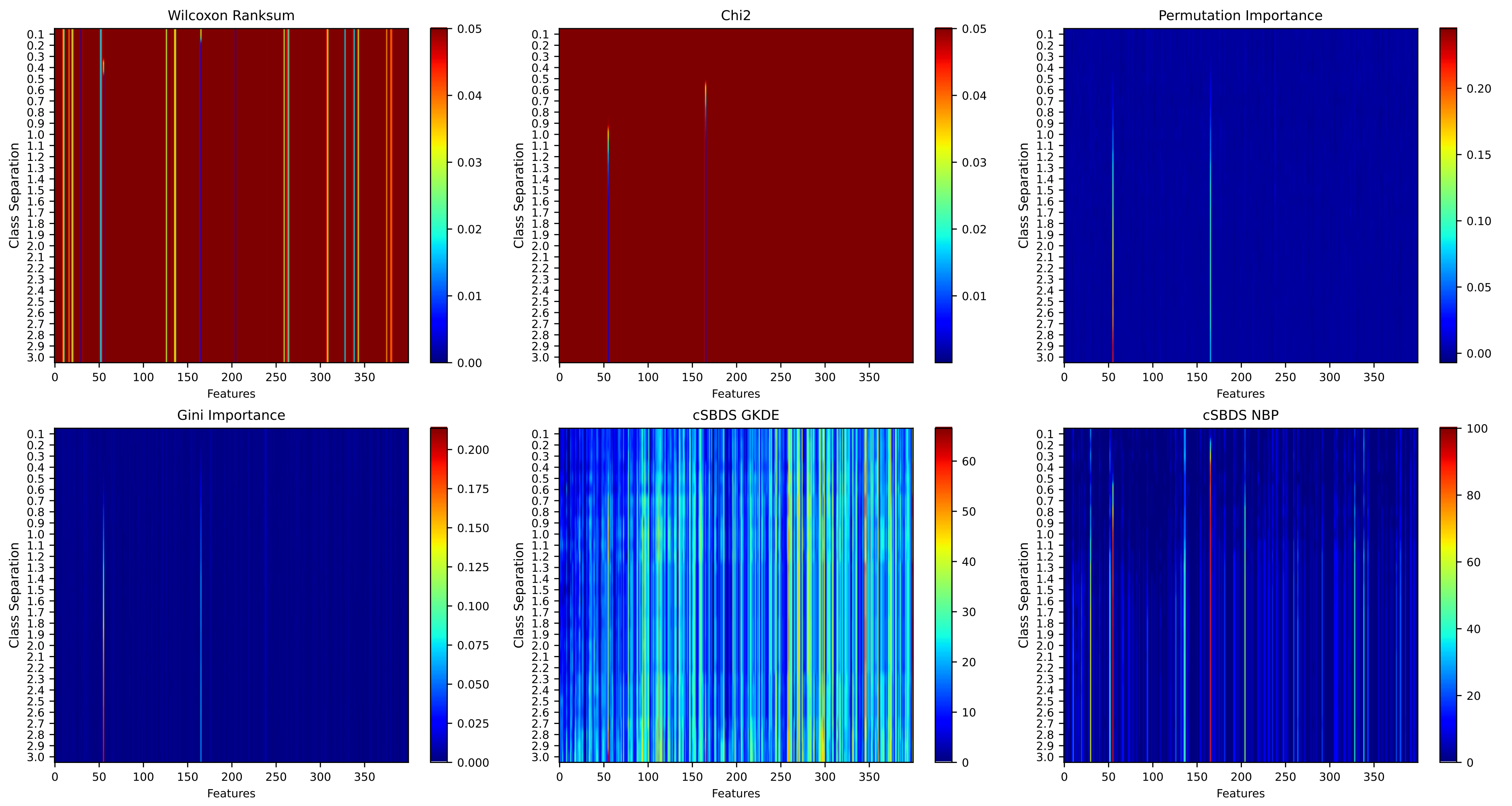


Figure 5.8: Feature importance methods comparison using synthetically created datasets varying the class separation parameter from 0.1 to 3 and with 1 cluster per class. For the Wilcoxon ranksum test and the chi-square test the p-value is shown (the p-value higher than 0.05 was set to 0.05 to simplify the visualisation). The cSBDS GKDE and the cSBDS NBP show the frequency of appearance of the features in the subspaces selected for the classification.

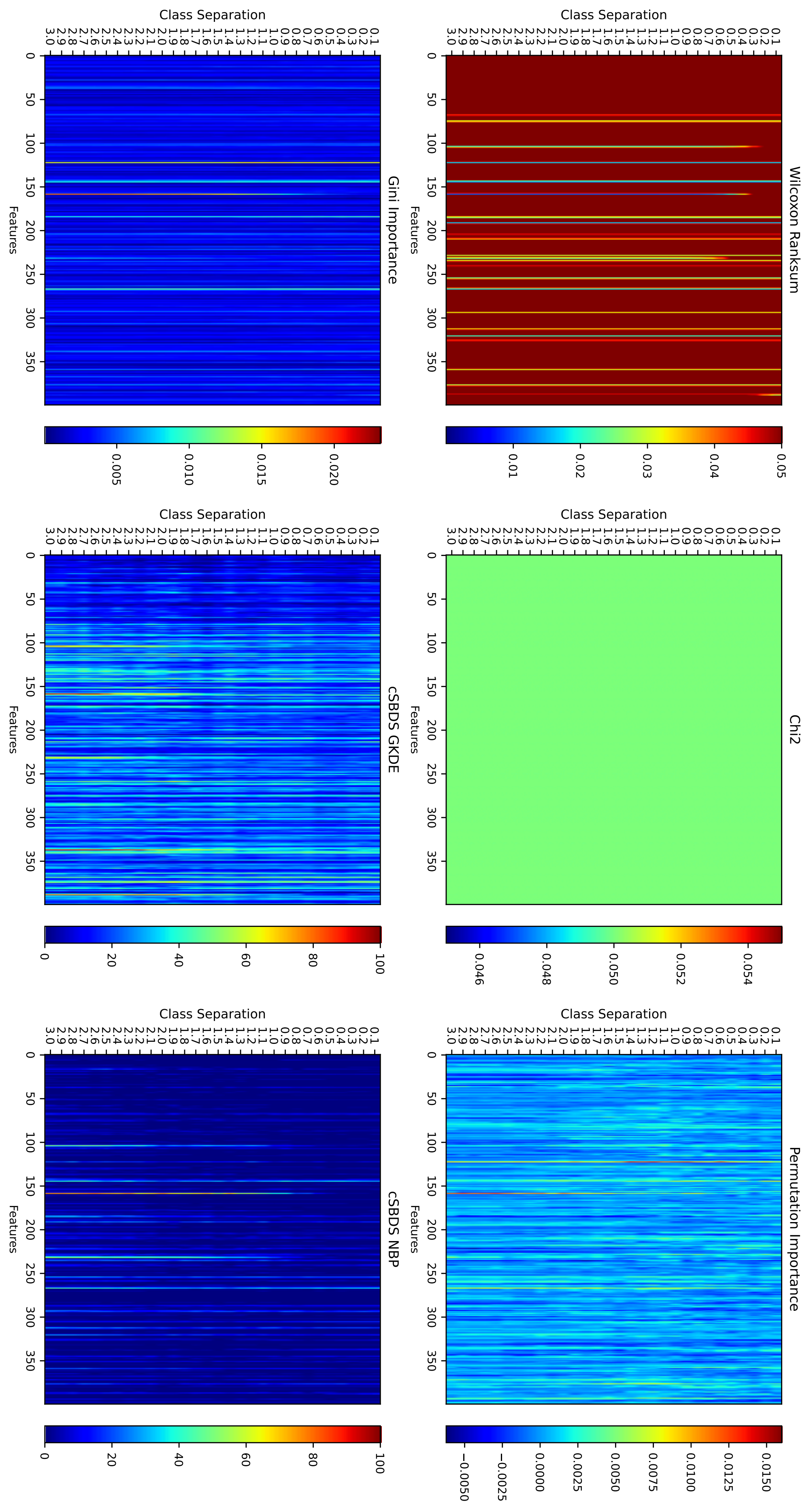


Figure 5.9: Feature importance methods comparison using synthetically created datasets varying the class separation parameter from 0.1 to 3 and with 10 cluster per class. For the Wilcoxon ranksum test and the chi-square test the p-value is shown (the p-value higher than 0.05 was set to 0.05 to simplify the visualisation). The cSBDS GKDE and the cSBDS NBP show the frequency of appearance of the features in the subspaces selected for the classification.

5.5 Conclusion

In this chapter, we investigated the feature selection approach embedded into the subspace clustering technique which is one of the core points of the Classifier SBDS (cSBDS) framework. First, we compared performance of the cSBDS framework with an RBF-SVM on the whole set of features and using a Wilcoxon rank sum test as a filter feature selection method (RBF-SVM FS). Three hundred synthetic datasets were created varying the number of clusters per class from 1 to 10 and the classification difficulty (class separation) from 0.1 to 3. The results indicated that the RBF-SVM FS performed statistically equivalent to the cSBDS GKDE with a confidence level of 99.999% on the majority of datasets based on the accuracy, sensitivity and specificity performances. Moreover, based on the datasets which had a statistically different performance, the RBF-SVM FS was usually the one with the better performance when compared to the cSBDS GKDE. This initially indicated that for these datasets, which are created based on Gaussian data, the use of the RBF-SVM FS was preferable.

Therefore, we hypothesised that a change on how the one-dimensional clusters are selected could improve the classification of the cSBDS framework. The Jenks Natural Break Points (NBP) method was proposed to substitute the Gaussian Kernel Density Estimator (GKDE) method. The results indicated that the cSBDS NBP was either statistically equivalent or statistically better in the whole majority of cases when compared with the RBF-SVM, RBF-SVM FS and the cSBDS GKDE. This indicates that the proposed substitution was beneficial for analysing these datasets and to select the most important features for them.

Next, we analysed three feature importance techniques: Gini, permutation and statistical. Three hundred synthetic datasets were used to compare the feature importance methods and the cSBDS methods (GKDE and NBP). The results indicate that when the number of clusters per class increase, the feature importance techniques lose their capability of indicating which are the most important features with the importance of the best ranked feature being closer to the importance of the worst features. The cSBDS NBP method was still able to indicate the most important features regardless of the number of clusters per class, showcasing the advantage of this method when a dataset have multiple clusters per class.

Therefore, the methods proposed in this chapter attempt to explain first for each type of datasets the cSBDS framework performs better when compared to the SBDS and a RBF-SVM before and after a feature selection approach. To do this, multiple datasets were created varying the classification difficulty and including an increasing number of clusters per class in the data. The results indicated that the SBDS framework have a statistically better performance on datasets with a higher classification difficulty and higher number of clusters per class. Second, multiple feature selection and importance methods were compared with the cSBDS framework on synthetic data with a known set of relevant features and a majority of noisy features. This analysis is important to evaluate if the subspace clustering method is able to select the most important features for the cSBDS framework. The results indicate that this is the case and that the cSBDS is able to uncover the most important features on datasets with different classification difficulties and number of clusters per class.

Chapter 6

Conclusions and Future Work

***

6.1 Thesis summary

The core aim of this thesis has been to provide a novel framework to evaluate the performance of small instance high-dimensional datasets. This framework was based on the dynamic selection of classifiers theory which was combined with a subspace clustering approach to overcome issues related to high-dimensional feature spaces. Fitting this framework on synthetic and real-world datasets allowed the development of the algorithm improving it on different characteristics of the datasets.

The thesis first explored how dynamic selection methods perform on small instance high-dimensional subspaces, this led to an understanding in Chapter 3 that these methods do not perform statistically better than the majority voting algorithm. The reason for this was that the majority of dynamic selection methods are based on a k-NN to define the region of competence. In Chapter 3 the dynamic selection methods were evaluated first on synthetic data and the increase in the number of features led to a decrease in the performance of the algorithms. Next, they were compared with majority voting on real-world datasets and their performance was found to be statistically equivalent, which indicates that modifications needed to be done in the framework of DS methods to improve their performance.

Chapter 4 presented the Subspace-Based Dynamic Selection (SBDS) framework, a novel method to evaluate small instance high-dimensional datasets. This method is based on the dynamic selection framework. However, to overcome the difficulties of high-dimensional datasets, a subspace clustering approach was proposed to replace the k-NN in the definition of the region of competence. It was shown that this modification was able to improve the performance when compared to DS methods. Nonetheless, the SBDS framework presented 3 important limitations that needed to be addressed: (1) the framework was initially tested in a single dataset, and for this dataset, the accuracy was improved but the sensitivity and specificity were reduced when compared to other DS methods; (2) the subspaces were found by using an unsupervised method, i.e. using the samples from both classes; and (3) the framework used a multiplier factor that was found empirically for the studied dataset. Therefore, we proposed the Classifier SBDS (cSBDS) framework which overcomes the aforementioned limitations and was found to be statistically better than other DS methods and majority voting. Moreover, the changes made to how the cSBDS finds the subspaces, allowed the cSBDS to select fewer features and more subspaces when compared to the SBDS framework.

The cSBDS framework was capable of performing statistically better than the majority voting, but it still has another advantage when compared to the traditional DS methods, which is the capability to select the most important features. In Chapter 5, approaches to understanding how the feature selection aspect of the subspace clustering part of the cSBDS framework compares with other feature selection methods are discussed. By using synthetically created Gaussian data, we found that the proposed subspace clustering approach on the cSBDS was limited to differentiating informative Gaussian data from noise, due to the use of the Gaussian Kernel Density Estimator (GKDE). Therefore, the Jenks Natural Break Points (NBP) was proposed and for the synthetic data it was observed that it performed better than the GKDE method.

6.2 Conclusions

Several specific conclusions can be drawn from this thesis. One of the most important conclusions is that the researchers need to be careful when applying methods on small instance high-dimensional datasets. The special characteristics of these datasets can lead to a decrease in performance and issues when evaluating feature importance. The recent development of DS methods and their effectiveness in performing better than static ensemble methods on small instance datasets led to a belief that these methods could be a good starting point to analyse datasets that have more features than samples. Nonetheless, since the majority of DS methods are based on a k-NN to define the region of competence, modifications needed to be done to the DS framework to adapt it to deal with high-dimensional datasets.

The comparison of DS methods on synthetic data and with majority voting on real-world datasets led to the understating that the k-NN deteriorate its performance when the number of features increases. Moreover, DS methods that did not use a k-NN to define the region of competence were still statistically equivalent to majority voting, which indicated that not only hubness and distance concentration in high-dimensional datasets can be a problem, but also that these datasets have peculiarities that needed to be addressed as well. One of the most important is that some features might only work for a subset of samples and appear as noise for the rest of the samples as indicated by Tian and Gu (2019) [87].

To deal with these problems, the SBDS framework proposed to incorporate a subspace clustering approach to define the region of competence. Nonetheless, the SBDS had some limitations that needed to be overcome. The modifications proposed led to the creation of the cSBDS framework which demonstrated that it can perform statistically better than majority voting and other DS methods. Furthermore, when compared to the SBDS, the cSBDS was able to select fewer features and more subspaces, making them more important for the classification which helped to increase the performance. It is important to note that both the SBDS and the cSBDS framework also have the advantage of indicating which are the most important feature for the classification since the subspace clustering part acts as a feature selection algorithm.

The feature selection analysis of the cSBDS framework also led to interesting findings. Firstly, the performance of the cSBDS was evaluated on multiple synthetic Gaussian data with an RBF-SVM with and without a filter feature selection approach. This analysis indicated that the GKDE part of the subspace clustering did not perform rightly. This happened because both the information and the useless features were drawn from Gaussian distributions. Consequently, the substitution of the GKDE by the NBP improved the performance due to a better selection of the subspaces. Furthermore, by comparing it with traditional feature selection methods the results indicated that it was able to select the most important features better than traditional methods on the tested synthetic datasets.

6.3 Limitations

As discussed in Chapter 4 the SBDS approach had two important limitations that were overcome for the proposal of the cSBDS framework. Nonetheless, both frameworks have important limitations concerning the generation time of subspaces. For very large data in terms of features and samples, the method proposed here would become intractable.

Furthermore, when evaluating the performance of the cSBDS, it is important to point out that the type of data influence the choice of how subspaces can be selected, as expected and shown on Chapter 5. Therefore, studies to understand better the characteristics of the data must be conducted in advance to decide which cSBDS framework should be applied: the one with a GKDE or the one with an NBP to define the one-dimensional clusters.

The bottleneck of the SBDS and cSBDS is the subspace clustering method, due to its high computational cost. The method proposed here was inspired by the work of Tian and Gu (2019) [87]. This approach can be classified into a cell-based approach and find subspaces efficiently. According to Parsons *et al.* (2004) [10], since there is no universal definition of clustering, there is also no universal definition of measures to compare clustering results. This is a great limitation of subspace clustering and clustering itself since the cluster quality measures are heuristics and do not guarantee meaningful results, hence the clusters found should be verified by domain experts. In high-dimensional feature spaces, the number of possible subspaces can be huge, requiring efficient search algorithms, which can be biased and greatly affect the assumptions made by the chosen algorithm. The SBDS and cSBDS use a bottom-up approach which has the advantage of reducing the search space, but because the pruning happens earlier the subspace detection the accuracy is influenced, which is a big limitation of these methods.

6.4 Contributions

This section summarises the core major and minor contributions to knowledge presented in this thesis.

6.4.1 Major contributions

Dynamic selection methods evaluation on high-dimensional datasets

Chapter 3 presented a comparative analysis between different dynamic selection methods and majority voting on small instance high-dimensional datasets. The results showed that the performance decreases when the number of dimensions increases, and most importantly that DS methods are statistically equivalent to the majority voting classifier. This has an implication in the area of ensemble learning since DS are considered one of the most important areas for multiple classifier systems recently due to its higher performance when compared to single classifiers and static ensembles. Nonetheless, before this thesis and the work carried out during the project, DS methods were not tested on small instance high-dimensional datasets.

Subspace-Based Dynamic Selection (SBDS)

The most significant theoretical contribution of this thesis is the Subspace-Based Dynamic Selection (SBDS) framework, which provides a novel DS framework combined with a subspace clustering approach to define the regions of competence. The SBDS framework aims to have a better performance when compared to dynamic selection methods. The main difference between the SBDS over dynamic selection is the use of subspace clustering to search through the feature and sample spaces for relevant clusters. Moreover, since the subspace clustering method gives the best features for a specific group of samples, we are able to further understand their importance. Chapter 4 describes in detail the framework and shows how it performs on a protein microarray dataset compared to other dynamic selection methods. The results indicated that the SBDS achieved the highest accuracy when compared to the DS methods investigated, but did not achieve the highest sensitivity or specificity.

Classifier SBDS (cSBDS)

Due to two clear limitations of the SBDS framework, we proposed the cSBDS framework. The main limitations of the SBDS framework are: (1) the subspaces were found by using samples from both classes, (2) the multiplier factor empirically found to define the distance between subspace to help on the nearest subspace search. Chapter 4 describes in detail how these limitations were overcome and which changes were made to the initial SBDS framework. When analysing the cSBDS on real-world small instance high-dimensional datasets, the results indicated that the cSBDS was able to overcome the limitations of the SBDS framework and most importantly, when compared to the majority voting classifier and other dynamic selection, the cSBDS was able to perform statistically better than them. Therefore, the cSBDS framework was the most significant practical contribution of this thesis.

6.4.2 Minor contributions

Novel subspace clustering methodology

Chapter 4 proposes a novel subspace clustering approach that is part of the SBDS and cSBDS frameworks. The method was inspired by Tian and Gu (2019) [87] and it is formed by two steps: (1) one-dimensional clustering, and (2) merging procedure. The merging procedure is the same one proposed by Tian and Gu (2019) [87], whilst the one-dimensional clustering approach substitutes the self-organising maps proposed by the authors. To find one-dimensional clusters this thesis proposes two approaches: the Gaussian Kernel Density Estimator (GKDE) in Chapter 4 and the Jenks Natural Break Points (NBP) in Chapter 5. The results in these two chapters indicate that the method to find one-dimensional clusters depends on the type of data studied, when using Gaussian datasets it is preferable to use a NBP approach to avoid bias in the selection of subspaces.

Feature selection analysis of the cSBDS framework

Chapter 5 investigated the feature selection approach embedded into the cSBDS framework. The proposed cSBDS with a Jenks Natural Break Points (NBP) to find the one-dimensional clusters was presented. This method performed statistically better when compared to a Radial Basis Function Support Vector Machine (RBF-SVM) classifier with a filter feature selection approach. Moreover, the cSBDS NBP was able to correctly indicate which were the most important features on datasets with a high number of clusters per class, while the investigated feature importance methods show a decrease in the difference between the most important features and the worst feature.

6.5 Future work

The following section reviews some of the potential directions for future work, specially in relation to overcome the limitations described in this work.

Improvements for computational cost The methods proposed in this thesis are computationally expensive since they are based on a subspace clustering approach and used on high-dimensional datasets. As the number of features and samples increase, the number of possible combinations to find subspaces increases as well. Therefore, parallel computing methods should be investigated in order to speed up the frameworks.

Pool generation and diversity When considering the use of DS, a diverse pool of classifiers should be targeted, because, intuitively, a diverse pool of classifiers means that different classifiers will be specialised in different regions of the feature space. However, diversity, is still an open question in the DS area and it should be more carefully studied. The methods proposed in this thesis (SBDS and cSBDS) generate intrinsically diverse classifiers, since each classifier is trained only in one specific region of the feature space, determined by the subspace clustering method. Nonetheless, future work on increasing the diversity by using different classifiers in different regions could perhaps help to improve the performance of these two methods. Moreover, a RSS approach should be used to better compare the performance between DS methods and the proposed SBDS and cSBDS, as well as using the RSS approach to define the subspaces in these two methods to verify which features are selected and the chances in performance.

Use of other subspace clustering approaches The subspace clustering method is the bottleneck in terms of the computational power required to run the proposed methods. In this thesis, the proposed subspace clustering method was based on the work done by Tian and Gu [87] with a one-dimensional cluster approach. This method is a bottom-up approach which tends to be less computational expensive. Nonetheless, studies on the use of top-down subspace methods need to be conducted to investigate if other methods are preferable depending on the studied data.

Further analysis to understand the type of data Results in this thesis appear to indicate that the type of data influence the selection of the one-dimensional clustering approach on the cSBDS framework. The work presented here is not fully conclusive. Further analytical and statistical methods should be used to test which is the best type of data for each one-dimensional clustering approach. If so, pruning methods could be developed for the subspace clustering approach to improve the subspace search based on specific data types or use other methods to find one-dimensional clusters.

Data complexity measures a better understanding of the data and classifiers dependencies is crucial to understand which classification methods should be used to analyse different problems. Such understanding is not simple and data complexity measures focused on the geometrical characteristics of the data class distributions could be used, not only the statistics or information theoretical descriptions. Therefore, measures that can highlight the manner in which classes are separated or interleaved could be used in a future work to decide which subspace clustering approach should be used and which classification methods should be used in the SBDS and cSBDS frameworks.

Appendices

Appendix A

Dynamic Selection Algorithms

***

The following sections will describe in details the different methods for dynamic selection. Also, the following mathematical notation is used to describe the different concepts comprised in the dynamic selection approaches.

- $\mathbb{L}$ is the test set of unknown samples;
- $\mathbb{V}$ is the validation set where the RofC will be computed;
- $\mathbb{T}$ is the training set where the pool of classifiers will be trained;
- $C = \{c_1, c_2, \ldots, c_N\}$ is the pool of classifiers consisting of N base classifiers;
- $\Omega = \{\omega_1, \ldots, \omega_L\}$ is the set of L classes in the classification problem
- ω_l is the class predicted by the classifier c_i;
- ω_t is the correct class of sample x;
- $\mathbf{x}_{j,test}$ is the test sample with unknown class label;
- $\theta_j = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K\}$ is the RofC of $\mathbf{x}_{j,test}$, $\mathbf{x}_k$ is one instance belonging to θ_j and K is the number of samples;
- $P(\omega_t \mid \mathbf{x}_k \subset \theta_j, c_i)$ is the posteriori probability of class ω_t provided by classifier c_i given a pattern $\mathbf{x}_k \in \omega_t$ belonging to θ_j
- $P(c_i(\mathbf{x}_{j,test}) = \omega_l \mid \mathbf{x}_k, c_i)$ is the posterior probability of class $c_i(\mathbf{x}_{j,test}) = \omega_l$ provided by the classifier c_i given a pattern $\mathbf{x}_k$ belonging to θ_j;
- $W_k = \dfrac{1}{d_k}$ and d_k is the Euclidean distance between the test sample $\mathbf{x}_{j,test}$ and its neighbour sample $\mathbf{x}_k$;
- $\delta_{i,j}$ is the estimated competence of the base classifier c_i for the test sample $\mathbf{x}_{j,test}$

A.1 History of DS methods

In 1993, Sabourin *et al.* [58] introduced the first DCS approach based on the rank of each classifier - Classifier Rank (CR). In 1997, Woods *et al.* [59] introduced: (1) a modification of CR named Modified Classifier Rank (MCR); and (2) a DCS method based on the local classification accuracy known as DCS by Local Accuracy (DCS-LA), with two methods to estimate the local accuracy: the Overall Local Accuracy (OLA) and the Local Class Accuracy (LCA). Two years later, Giacinto & Roli [60] proposed a modification over the OLA and LCA methods by considering the estimates of the class posterior probabilities, creating two methods: *a Priori* and *a Posteriori*. The same authors [61] proposed the Multiple Classifier Behaviour (MCB), which exploits the concept of Behaviour Knowledge Space (BKS). Smits [62], in 2002, also proposed a modification on the DCS-LA method by using distance weights to overcome outliers in the region of competence; this methods is known as Modified Local Accuracy (MLA). Recently in 2016, Brun *et al.* [69] proposed the Dynamic Selection on Complexity (DSOC) method that considers not only the accuracy of each classifier in the region of competence, but also the use of features related to the problem complexity.

Other authors focused on developing methods of DS that result in selecting an ensemble of classifiers (DES). By selecting an ensemble of classifiers, the DES methods can overcome one of the most critical points of DCS methods: the choice of one individual classifier relies on how much we trust the estimate of generalization of the classifiers. In other words, selecting a single classifier can be highly error-prone. On the other hand, if an ensemble of classifiers is selected the risk of this over-generalization is reduced [3, 4, 7]. Another reason for selecting an ensemble of classifiers is that, frequently, some base classifiers can have the same competence level and instead of selecting a random one it is more reasonable selecting all of them [4].

In 2006 and 2008, Soares *et al.* [63] and Souto *et al.* [64] proposed two DES methods based on accuracy and diversity to select classifiers using a k-NN (DES-kNN) and a k-Means (DES-kMeans) to find the RofC. Also in 2008, Ko *et al.* [7] proposed the class of methods k-Nearest ORAcles (KNORA), which are based on the concept of the *Oracle*. The *Oracle* is defined as the ideal select for a pool of classifiers, i.e., it will always correctly classifies a test sample if at least one classifier in the pool correctly classify the test sample; hence, this is considered the upper bound of a MCS. The KNORA methods attempt to find local oracles in the RofC.

Between 2009 and 2012, Antosik & Kurzynski [66], Woloszynski & Kurzynski [29, 65] and Woloszynski *et al.* [67] proposed four DES methods based on a probabilist model.

By calculating the competence of each classifier with respect to a random guessing, these methods gained a meaningful interpretation: competent (incompetent) classifier are more (less) accurate than the random classifier. All classifiers that achieve a competence higher than the random classifier are selected to form the ensemble to predict the label of the unknown test sample. In 2012, Woloszynski *et al.* [67], also proposed a method based on random classification (DES-P), which finds the region of competence using a k-NN; and from this samples, estimates the accuracy of each classifier; then, all classifiers that achieve a competence higher than the random guessing are selected to form the ensemble.

In 2013, Cavalin *et al.* [68] proposed the k-Nearest Output Profiles (KNOP) methods which selects the RofC based on the decision space, i.e., all samples in the test set and validation set are transformed into output profiles. The output profile of an instance $\tilde{\mathbf{x}}_j$ is denoted by $\tilde{\mathbf{x}}_j = \{\tilde{\mathbf{x}}_{j,1}, \tilde{\mathbf{x}}_{j,2}, \ldots, \tilde{\mathbf{x}}_{j,M}\}$, where $\tilde{\mathbf{x}}_{j,i}$ is the predicted label of $\mathbf{x}_j$ by the i-th classifier.

Recently, in 2016, Cruz *et al.* [30] proposed the Meta-learning - DES (Meta-DES) method that uses meta-learning and five distinct meta-features, each one corresponding to a different criterion to measure the competence of a base classifier, to predict the label of an unknown test sample.

A.2 Classifier Rank (CR)

Sabourin *et al.* [58] presented this algorithm which selects a classifier c_i based on the number of consecutive neighbouring samples correctly classified by c_i. The selected classifier is said to have the highest "rank".

A.3 Modified Classifier Rank (MCR)

Woods *et al.* [59] presented an alternative for the CR method. Given a test sample assigned to class ω_l by a classifier c_i, the competence of c_i is the number of consecutive neighbouring samples assigned to class ω_l by c_i that were correctly labelled by c_i.

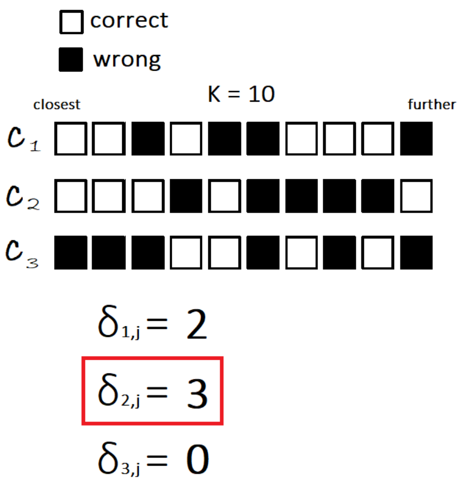


Figure A.1: Example of CR method. The multiple classifier system is formed by three classifiers $\{c_1, c_2, c_3\}$ and a RofC using a 10-Nearest Neighbour classifier. In this example, the classifier c_2 is the one chosen to classify the test pattern $\mathbf{x}_{j,test}$.

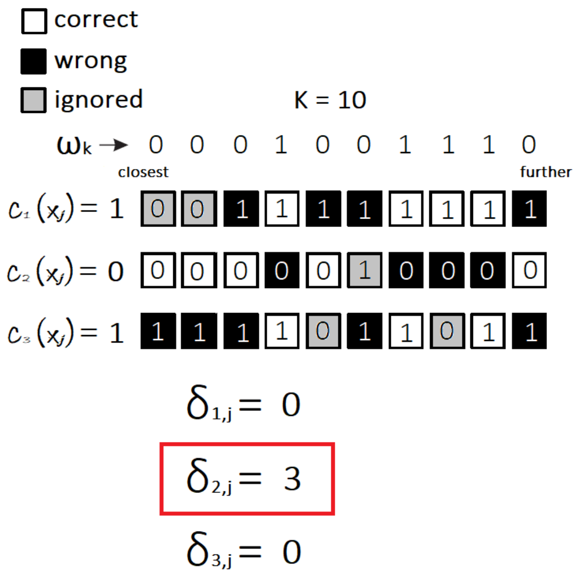


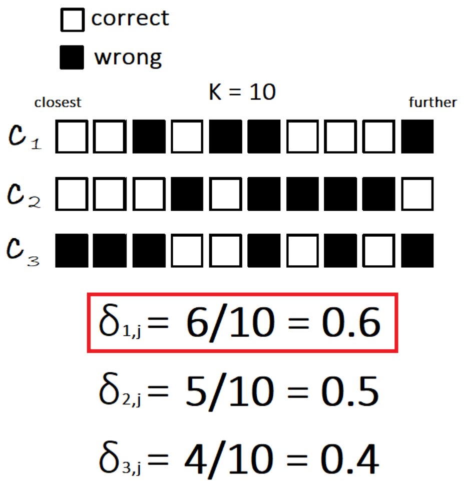
Figure A.2: Example of MCR method. The multiple classifier system is formed by three classifiers $\{c_1, c_2, c_3\}$ and a RofC using a 10-Nearest Neighbour classifier. The three classifiers predict the label of test sample $\mathbf{x}_{j,test}$ and their ranks are calculated based on the class predicted to $\mathbf{x}_{j,test}$. In this example, the classifier c_2 is the one chosen to classify the test pattern $\mathbf{x}_{j,test}$.

A.4 DCS by Local Accuracy (DCS-LA)

This method was proposed by Woods *et al.* [59] and it is based on local accuracy estimates. The basic idea is to estimate the competence of each base classified using its accuracy in local regions of the feature space surrounding the test sample. These local regions are defined using a k-NN. The authors propose two methods to estimate the local accuracy:

Algorithm 3: OLA / LCA

```
input  : $\mathbb{L}$, $\mathbb{V}$, $\mathbb{T}$ and $C$
output: $\upsilon$
/* $\upsilon$ is the array with the predicted label for all samples in $\mathbb{L}$ */
foreach $\mathbf{x}_{j,test} \in \mathbb{T}$ do
    if $\forall c_i \in C$ predict the same label $\omega_l$ for $\mathbf{x}_{j,test}$ then
        $\upsilon_j = \omega_l$ ;
    else
        Find the RofC $\theta_j$ of $\mathbf{x}_{j,test}$ using a k-NN in the validation set $\mathbb{V}$ ;
        foreach $c_i \in C$ do
            Calculate $\delta_{i,j}$ using equation A.1 (OLA) or A.2 (LCA) ;
        end
        Select the classifier $c_t^*$ with the highest $\delta_{i,j}$ ;
        $\upsilon_j = c_t^*(\mathbf{x}_{j,test})$ ;
    end
end
```



□ correct
■ wrong
K = 10
closest
further
c_1
c_2
c_3
$\delta_{1,j} = 6/10 = 0.6$
$\delta_{2,j} = 5/10 = 0.5$
$\delta_{3,j} = 4/10 = 0.4$

Figure A.3: Example of OLA method. The multiple classifier system is formed by three classifiers $\{c_1, c_2, c_3\}$ and a RofC using a 10-Nearest Neighbour classifier. In this example, the classifier c_1 is the one chosen to classify the test pattern $\mathbf{x}_{j,test}$.

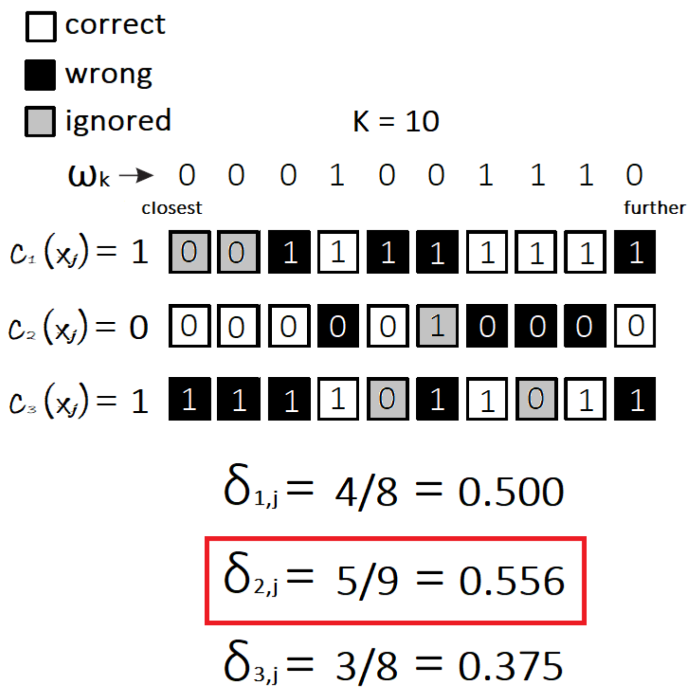


Figure A.4: Example of LCA method. The multiple classifier system is formed by three classifiers $\{c_1, c_2, c_3\}$ and a RofC using a 10-Nearest Neighbour classifier. The three classifiers predict the label of test sample $\mathbf{x}_{j,test}$ and their accuracies are calculated based on the class predicted to $\mathbf{x}_{j,test}$. In this example, the classifier c_2 is the one chosen to classify the test pattern $\mathbf{x}_{j,test}$.

Overall Local Accuracy (OLA) is the percentage of neighbouring samples of the test sample $\mathbf{x}_{j,test}$ that are correctly classified.

$$\delta_{i,j} = \frac{N_i}{K} \tag{A.1}$$

where N_i is the number of neighbouring samples of $\mathbf{x}_{j,test}$ that are correctly classified by c_i and K is the size of the RofC.

Local Class Accuracy (LCA) exploits the information that ω_l is the class assigned by the classifier c_i to the test pattern $\mathbf{x}_{j,test}$. Therefore, we can determine the percentage of neighbouring sample assigned to class ω_l by the classifier c_i that have been correctly labelled.

$$\delta_{i,j} = \frac{N_{i,l}}{N_{k,l}} \tag{A.2}$$

where $N_{i,l}$ is the number of neighbouring samples of $\mathbf{x}_{j,test}$ that have been correctly assigned by c_i with class ω_l; and $N_{k,l}$ is the total number of patterns in the RofC that have been assigned by c_i with class ω_l.

Algorithm 4: a Priori

```
input  : $\mathbb{L}$, $\mathbb{V}$, $\mathbb{T}$ and $C$
output: $\upsilon$
/* $\upsilon$ is the array with the predicted label for all samples in $\mathbb{L}$ */
1  foreach $\mathbf{x}_{j,test} \in \mathbb{T}$ do
2      if $\forall c_i \in C$ predict the same label $\omega_l$ for $\mathbf{x}_{j,test}$ then
3          $\upsilon_j = \omega_l$ ;
4      else
5          $\Psi = \varnothing$ ;
6          Find $\theta_j$ of $\mathbf{x}_{j,test}$ using a k-NN in the validation set $\mathbb{V}$ ;
7          foreach $c_i \in C$ do
8              Calculate $\delta_{i,j}$ using equation A.4 ;
9              if $\delta_{i,j} \geq 0.5$ then
10                 $\Psi = \Psi \cup c_i$
11         end
12         $\delta_m = \max(\delta_{i,j} \mid c_i \in \Psi)$ ;
13         $c_{\delta_m} = arg\max(\delta_{i,j} \mid c_i \in \Psi)$ ;
14         $selected = TRUE$ ;
15         foreach $c_i \in \Psi$ do
16             $d = \delta_m - \delta_{i,j}$ ;
17             if $(i \neq m)$ and $(d < Threshold)$ then
18                 $selected = FALSE$ ;
19         end
20         if $selected == TRUE$ then
21             $c_t^* = c_{\delta_m}$ ;
22         else
23             $c_t^*$ is randomly selected from $\Psi$ with $d < Threshold$ ;
24         end
25         $\upsilon_j = c_t^*(\mathbf{x}_{j,test})$ ;
26     end
27 end
```

A.5 A priori

Giacinto & Roli [60] proposed an adaptation for the method OLA. By choosing classifiers with the ability to provide estimates of the class posterior probability, the authors reformulated the equation A.1. Given a pattern $\mathbf{x}_k \in \omega_t$ belonging to the neighbourhood

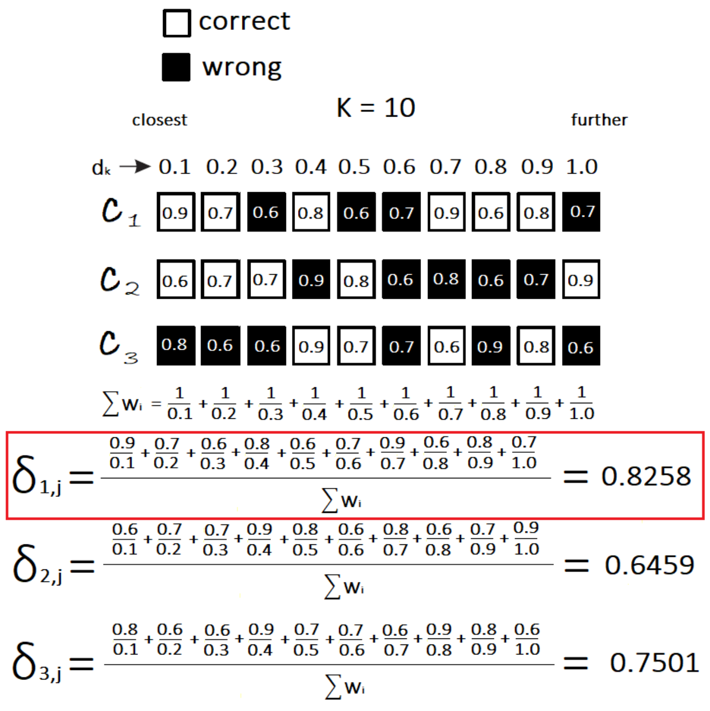
θ_j, the $P(\omega_t|\mathbf{x}_k \subset \theta_j, c_i)$ can be regarded as a measure of accuracy of the classifier c_i for the pattern $\mathbf{x}_k$. Therefore, the equation A.1 can be rewritten as follows:

$$\delta_{i,j} = \frac{1}{K}\sum_{k=1}^{K} P(\omega_t \mid \mathbf{x}_k \subset \theta_j, c_i) \tag{A.3}$$

According to equation A.4, the selection is performed without knowing the class predicted by the classifier c_i to the test pattern $\mathbf{x}_{j,test}$.

In order to handle the "uncertainty" in the definition of the RofC, each instance in θ_j is weighted by the inverse of the Euclidean distance d_k of patterns $\mathbf{x}_k$ to $\mathbf{x}_{j,test}$:

$$\delta_{i,j} = \frac{\sum_{k=1}^{K} P(\omega_t \mid \mathbf{x}_k \subset \theta_j, c_i) W_i}{\sum_{k=1}^{K} W_i} \tag{A.4}$$



□ correct
■ wrong
closest
K = 10
further
dk → 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
c1 0.9 0.7 0.6 0.8 0.6 0.7 0.9 0.6 0.8 0.7
c2 0.6 0.7 0.7 0.9 0.8 0.6 0.8 0.6 0.7 0.9
c3 0.8 0.6 0.6 0.9 0.7 0.7 0.6 0.9 0.8 0.6
∑wi = 1/0.1 + 1/0.2 + 1/0.3 + 1/0.4 + 1/0.5 + 1/0.6 + 1/0.7 + 1/0.8 + 1/0.9 + 1/1.0
δ1,j = (0.9/0.1 + 0.7/0.2 + 0.6/0.3 + 0.8/0.4 + 0.6/0.5 + 0.7/0.6 + 0.9/0.7 + 0.6/0.8 + 0.8/0.9 + 0.7/1.0) / ∑wi = 0.8258
δ2,j = (0.6/0.1 + 0.7/0.2 + 0.7/0.3 + 0.9/0.4 + 0.8/0.5 + 0.6/0.6 + 0.8/0.7 + 0.6/0.8 + 0.7/0.9 + 0.9/1.0) / ∑wi = 0.6459
δ3,j = (0.8/0.1 + 0.6/0.2 + 0.6/0.3 + 0.9/0.4 + 0.7/0.5 + 0.7/0.6 + 0.6/0.7 + 0.9/0.8 + 0.8/0.9 + 0.6/1.0) / ∑wi = 0.7501

Figure A.5: Example of a Priori method. The multiple classifier system is formed by three classifiers $\{c_1, c_2, c_3\}$ and a RofC using a 10-Nearest Neighbour classifier. The three classifiers estimate the posterior probability of each instance in θ_j according to its true label. Each instance is also weighted by the inverse of the Euclidean distance d_k of patterns $\mathbf{x}_k$ to $\mathbf{x}_{j,test}$. In this example, the classifier c_1 is the one chosen to classify the test pattern $\mathbf{x}_{j,test}$.

A.6 A posteriori

Giacinto & Roli [60] also proposed an adaptation for the method LCA. If the class assigned by the classifier c_i to the test pattern $\mathbf{x}_{j,test}$ is known ($c_i(\mathbf{x}_{j,test}) = \omega_l$), this information can be used to reformulate equation A.2. Analogously to the a Priori method, the "uncertainty" in the definition of $\theta_{j,test}$ is handled by weighting each posterior probability by the Euclidean distance d_k of patterns $\mathbf{x}_k$ to $\mathbf{x}_{j,test}$

$$\delta_{i,j} = \frac{\sum_{\mathbf{x}_k \in \omega_l} P(c_i(\mathbf{x}_{j,test}) = \omega_l \mid \mathbf{x}_k, c_i) W_i}{\sum_{k=1}^{K} P(c_i(\mathbf{x}_{j,test}) = \omega_l \mid \mathbf{x}_k, c_i) W_i} \qquad \text{(A.5)}$$

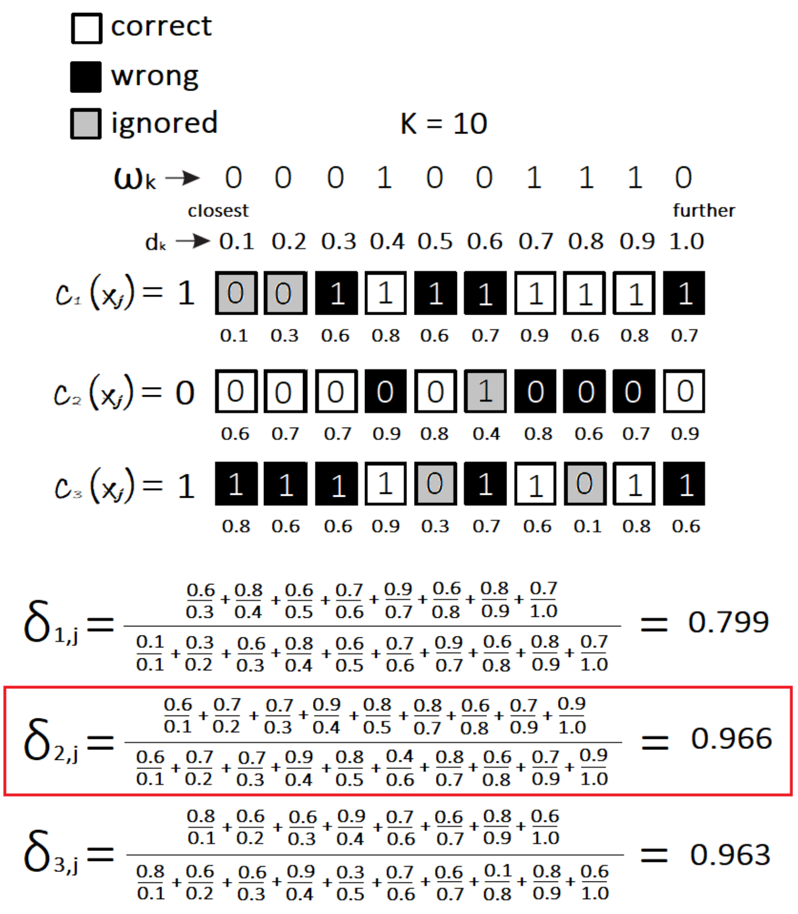


Figure A.6: Example of a Posteriori method. The multiple classifier system is formed by three classifiers $\{c_1, c_2, c_3\}$ and a RofC using a 10-Nearest Neighbour classifier. The three classifiers estimate the posterior probability of each instance in θ_j according to the label predict for $\mathbf{x}_{j,test}$. Each instance is also weighted by the inverse of the Euclidean distance d_k of patterns $\mathbf{x}_k$ to $\mathbf{x}_{j,test}$. In this example, the classifier c_2 is the one chosen to classify the test pattern $\mathbf{x}_{j,test}$.

Algorithm 5: a Posteriori

```
input  : $\mathbb{L}$, $\mathbb{V}$, $\mathbb{T}$ and $C$
output: $\upsilon$
/* $\upsilon$ is the array with the predicted label for all samples in $\mathbb{L}$ */
foreach $\mathbf{x}_{j,test} \in \mathbb{T}$ do
    if $\forall c_i \in C$ predict the same label $\omega_l$ for $\mathbf{x}_{j,test}$ then
        $\upsilon_j = \omega_l$ ;
    else
        $\Psi = \varnothing$ ;
        Find $\theta_j$ of $\mathbf{x}_{j,test}$ using a k-NN in the validation set $\mathbb{V}$ ;
        foreach $c_i \in C$ do
            Calculate $\delta_{i,j}$ using equation A.5 ;
            if $\delta_{i,j} \geq 0.5$ then
                $\Psi = \Psi \cup c_i$
        end
        $\delta_m = \max(\delta_{i,j} \mid c_i \in \Psi)$ ;
        $c_{\delta_m} = arg\max(\delta_{i,j} \mid c_i \in \Psi)$ ;
        $selected = TRUE$ ;
        foreach $c_i \in \Psi$ do
            $d = \delta_{max} - \delta_{i,j}$ ;
            if $(i \neq m)$ and $(d < Threshold)$ then
                $selected = FALSE$ ;
        end
        if $selected == TRUE$ then
            $c_t^* = c_m$ ;
        else
            foreach $c_i \in C$ do
                $conf_i(\mathbf{x}_{j,test}) = \min_{\substack{\rho=1,\ldots,L \\ \rho\neq l}} [P(\omega_l \mid \mathbf{x}_{j,test}, c_i) - P(\omega_\rho \mid \mathbf{x}_{j,test}, c_i)]$ ;
                $bel_i = conf_i(\mathbf{x}_{j,test}) \cdot \delta_{i,j}$ ;
                if $bel_i \geq 0.5$ then
                    $\hat{\Psi} = \hat{\Psi} \cup c_i$
            end
            $bel_m = \max(bel_i \mid c_i \in \hat{\Psi})$ ;
            $c_{bel_m} = arg\max(bel_i \mid c_i \in \hat{\Psi})$ ;
            $selected = TRUE$ ;
            foreach $c_i \in \hat{\Psi}$ do
                $d = bel_m - bel_i$ ;
                if $(i \neq m)$ and $(d < Threshold)$ then
                    $selected = FALSE$ ;
            end
            if $selected == TRUE$ then
                $c_t^* = c_{bel_m}$ ;
            else
                $c_t^*$ is randomly selected from $\hat{\Psi}$ with $d < Threshold$ ;
            end
        end
        $\upsilon_j = c_t^*(\mathbf{x}_{j,test})$ ;
    end
end
```

A.7 Multiple Classifier Behaviour (MCB)

Giacinto & Roli [61] proposed a new dynamic classifier selection based on classifier's local accuracy and multiple classifier behaviour. The MCB of a given pattern is a vector whose elements are the decisions (labels) predicted by each the individual classifier in the pool of classifiers, i.e., given x, the vector $MCB(\mathbf{x})$ is defined by $MCB(\mathbf{x}) = \{c_1(\mathbf{x}), c_2(\mathbf{x}), \dots, c_N(\mathbf{x})\}$.

For each unknown test pattern $\mathbf{x}_{j,test}$, θ_j is identified and the MCB is computed on $\mathbf{x}_{j,test}$ and all patterns on θ_j. Afterwards, the patterns on θ_j that satisfy the measure of similarity $S(\mathbf{x}_k, \mathbf{x}_{j,test}) >$ threshold are selected forming a new $\hat{\theta}_j$. Finally, the competence of each classifier is measured by the ratio between the number of patterns that were correctly classified by the classifier $c_i \in C$ and the total number of patterns in $\hat{\theta}_j$. The classifier with the highest competence is then selected. The similarity measure used to find $\hat{\theta}_j$ is defined as follows:

$$S(\mathbf{x}_k, \mathbf{x}_{j,test}) = \frac{1}{N} \sum_{i=1}^{N} T_i(\mathbf{x}_k, \mathbf{x}_{j,test}) \tag{A.6}$$

where $T_i(\mathbf{x}_k, \mathbf{x}_{j,test})$ is defined as:

$$T_i(\mathbf{x}_k, \mathbf{x}_{j,test}) = \begin{cases} 1, & \text{if } c_i(\mathbf{x}_k) = c_i(\mathbf{x}_{j,test}) \\ 0, & \text{if } c_i(\mathbf{x}_k) \neq c_i(\mathbf{x}_{j,test}) \end{cases} \tag{A.7}$$

It is worth noticing that this method has an advantage of not relying on the size of the neighbourhood to compute the RofC, because $\hat{\theta}_j$, which will be used to calculate the accuracy of each classifier, depends on the degree of similarity between the unknown test pattern and patterns in the region θ_j.

A.8 Modified Local Accuracy (MLA)

Proposed by Smits [62], the MLA DCS method aims to be more robust to the choice of size of the RofC. This technique assumes that, in the feature space, the classes maintain

Algorithm 6: MCB

```
input  : $\mathbb{L}$, $\mathbb{V}$, $\mathbb{T}$ and $C$
output: $\upsilon$
/* $\upsilon$ is the array with the predicted label for all samples in $\mathbb{L}$ */
foreach $\mathbf{x}_{j,test} \in \mathbb{T}$ do
    if $\forall c_i \in C$ predict the same label $\omega_l$ for $\mathbf{x}_{j,test}$ then
        $\upsilon_j = \omega_l$ ;
    else
        Find $\theta_j$ of $\mathbf{x}_{j,test}$ using a k-NN in the validation set $\mathbb{V}$ ;
        $\hat{\theta}_j = \varnothing$ ;
        foreach $\mathbf{x}_k \in \theta_j$ do
            Compute $S(\mathbf{x}_k, \mathbf{x}_{j,test})$ ;
            if $S(\mathbf{x}_k, \mathbf{x}_{j,test}) > Threshold$ then
                $\hat{\theta}_j = \hat{\theta}_j \cup \mathbf{x}_k$ ;
        end
        foreach $c_i \in C$ do
            Compute $\delta_{i,j}$ using equation A.1 with $\hat{\theta}_j$ ;
        end
        Select the classifier $c_t^*$ with the highest $\delta_{i,j}$ ;
        $\upsilon_j = c_t^*(\mathbf{x}_{j,test})$ ;
    end
end
```

a certain continuity. Therefore, neighbouring elements are expected to have a stronger relationship when compared to elements further away. In other works, when the size (K) of the RofC is too big, it might contain elements that can be considered as outliers, which will negatively influence the final competence of each classifier; however, when K is too small it may lead to insufficient information. Therefore, MLA weights each instance in θ_j by its instance to $\mathbf{x}_{j,test}$. To tackle this issue, the MLA algorithm weights each correctly assigned pattern in θ_j by c_i with class w_l:

$$\delta_{r,j} = \frac{\sum W_{r,k}}{N_{k,l}} \tag{A.8}$$

where $W_{r,k}$ is the weight applied to the r-th pattern that was correctly assigned by c_i with class w_l; and $N_{k,l}$ is the total number of patterns in the RofC that have been assigned by c_i with class ω_l.

The classifier with the highest competence is selected to predict the label of $\mathbf{x}_{j,test}$. Three different weighting schemes were proposed in the literature:

1. **Dudani's weighting scheme**: Smits [62] proposed a MLA algorithm using the Dudani's weighting scheme. This scheme was proposed by Dudani [113] as a decision rule to improve the classification of a k-NN; it calculates the weight of the r-th of the k nearest neighbours based on the distance d_r between the unknown test pattern $\mathbf{x}_{j,test}$ and its r-th neighbour in the RofC.

$$W_{r,k} = \begin{cases} \dfrac{d_k - d_r}{d_k - d_1}, & \text{if } d_i \neq d_1 \\ 1, & \text{otherwise} \end{cases} \tag{A.9}$$

2. **Macleod's weighting scheme**: Smits [62] also proposed another MLA algorithm using now the Macleod's weighting scheme, which is a generalized version of the Dudani's weighting scheme proposed by Macleod *et al.* [114]. This scheme introduces a mechanism for scaling the distance using a user-specified parameter (α) and the s-th nearest neighbour. Good classification results were obtained by Smits [62] using $\alpha = 0$ and s set to the total number of samples in the validation set and $s = 3k$ where k is the size of the RofC.

$$W_{r,k} = \begin{cases} \dfrac{d_k - d_r + \alpha(d_s - d_1)}{(1+\alpha)(d_s - d_1)}, & \text{if } d_s \neq d_1 \\ 1, & \text{otherwise} \end{cases} \tag{A.10}$$

3. **Euclidean's weighting scheme**: Cruz *et al.* [4] proposed a MLA algorithm using the inverse of the Euclidean distance (d) between the pattern $\mathbf{x}_r \in \theta_j$ and the unknown test pattern $\mathbf{x}_{j,test}$.

$$W_{r,k} = \frac{1}{d(x_r \in \theta_j, \mathbf{x}_{j,test})} \tag{A.11}$$

where $W_{k,l}$ is the inverse of the euclidean distance between the unknown test pattern $\mathbf{x}_{j,test}$ and the pattern $\mathbf{x}_k \in \theta_j$.

Algorithm 7: MLA

```
input  : $\mathbb{L}$, $\mathbb{V}$, $\mathbb{T}$ and $C$
output: $\upsilon$
/* $\upsilon$ is the array with the predicted label for all samples in $\mathbb{L}$ */
1  foreach $\mathbf{x}_{j,test} \in \mathbb{T}$ do
2      if $\forall c_i \in C$ predict the same label $\omega_l$ for $\mathbf{x}_{j,test}$ then
3          $\upsilon_j = \omega_l$ ;
4      else
5          Find $\theta_j$ of $\mathbf{x}_{j,test}$ using a k-NN in the validation set $\mathbb{V}$ ;
6          foreach $c_i \in C$ do
7              Calculate $\delta_{i,j}$ using equation A.8 with one of the possible weighting
                 schemes: equation A.9 (Dudani) or A.10 (Macleod) or A.11
                 (Euclidean) ;
8          end
9          Select the classifier $c_t^*$ with the highest $\delta_{i,j}$ ;
10         $\upsilon_j = c_t^*(\mathbf{x}_{j,test})$ ;
11     end
12 end
```

A.9 DES - K-Nearest Neighbour (DES-kNN) and DES - k-Means (DES-kMeans)

Soares *et al.* [63] proposed a dynamic ensemble selection method which considers both the accuracy and diversity of the classifiers in the RofC to select an ensemble of classifiers to predict the label of the unknown test pattern $\mathbf{x}_{j,test}$. The diversity measure is used due to the fact that an ideal situation for any multiple classifier system is to have a set of classifiers with uncorrelated or negatively correlated errors, i.e., this classifiers should make different errors, being, therefore, diverse. To do this, two different versions to define the RofC were presented: the first one uses a k-NN (DES-kNN), while the second uses a kMeans clustering algorithm (DES-kMeans). In both cases, the accuracy and diversity of each classifier $c_i \in C$ is calculated based on the RofC provided for this methods for each unknown test pattern.

The accuracy of each classifier in the RofC is computed as the proportion of neighbours in which the classifier $c_i \in C$ has provided the true label. On the other hand, the diversity is computed using the double-fault measure (Equation A.12). This measure uses

Algorithm 8: DES-kMeans

```
input  : $\mathbb{L}$, $\mathbb{V}$, $\mathbb{T}$ and $C$
output: $\upsilon$
/* $\upsilon$ is the array with the predicted label for all samples in $\mathbb{L}$ */
1  Cluster via $k$Means the patterns in $\mathbb{V}$ into $k$ clusters ;
2  For each cluster produced, rank the classifiers in $C$ in a decreasing order of
     accuracy and increasing order of diversity ;
3  foreach $\mathbf{x}_{j,test} \in \mathbb{T}$ do
4      if $\forall c_i \in C$ predict the same label $\omega_l$ for $\mathbf{x}_{j,test}$ then
5          $\upsilon_j = \omega_l$ ;
6      else
7          Assign $\mathbf{x}_{j,test}$ to the cluster that presents the nearest centroid (Euclidean
             distance) ;
8          Choose the $N$ most accurate classifiers of this cluster ;
9          From the $N$ most accurate, choose the $J$ most diverse classifier (the ones
             with the lowest $DF_{i,k}$ values) to compose the ensemble $\Psi$, where $J \leqslant N$ ;
10         $\upsilon_j = MajorityVote(\mathbf{x}_{j,test}, \Psi)$ ;
11     end
12 end
```

Algorithm 9: DES-kNN

```
input  : $\mathbb{L}$, $\mathbb{V}$, $\mathbb{T}$ and $C$
output: $\upsilon$
/* $\upsilon$ is the array with the predicted label for all samples in $\mathbb{L}$ */
1  foreach $\mathbf{x}_{j,test} \in \mathbb{T}$ do
2      if $\forall c_i \in C$ predict the same label $\omega_l$ for $\mathbf{x}_{j,test}$ then
3          $\upsilon_j = \omega_l$ ;
4      else
5          Find $\theta_j$ of $\mathbf{x}_{j,test}$ using a k-NN in the validation set $\mathbb{V}$ ;
6          Based on $\theta_j$, rank the classifiers in $C$ in a decreasing order of accuracy and
             increasing order of diversity ;
7          Select the $N$ most accurate classifiers of $\theta_j$ ;
8          From the $N$ most accurate, choose the $J$ most diverse classifier (the ones
             with the lowest $DF_{i,k}$ values) to compose the ensemble $\Psi$, where $J \leqslant N$ ;
9          $\upsilon_j = MajorityVote(\mathbf{x}_{j,test}, \Psi)$ ;
10     end
11 end
```

the proportion of cases that has been misclassified by both classifiers, and is defined as follows:

$$DF_{i,k} = \frac{N^{00}}{N^{11} + N^{10} + N^{01} + N^{00}} \tag{A.12}$$

where N^{00} is the number of patterns that both classifiers misclassified; N^{01} is the number of patterns that the first classifier correctly classified and the second misclassified; N^{10} is the number of patterns that the first classifier misclassified and the second correctly classified; and N^{11} is the number of patterns that both classifiers correctly classified. $DF_{i,k} = 1$ shows that both classifiers are always wrong, while $DF_{i,k} = 0$ shows that both classifiers are always right. Therefore, a low $DF_{i,k}$ is desired for an ensemble.

Given a classification task, let $\mathbb{L}$, $\mathbb{V}$ and $\mathbb{T}$ be three disjoint sets ($\mathbb{L} \cap \mathbb{V} \cap \mathbb{T} = \emptyset$) representing the training, validation and test sets. The two versions presented by Soares *et al.* [63] are presented on algorithms 8 and 9.

A.10 *k*-Nearest ORAcles (KNORA)

Ko *et al.* [7] proposed a method to find the most suitable ensemble of classifiers for each unknown test sample. The KNORA method also uses a k-NN to determine the neighbourhood of a test sample in a validation set. This method checks which classifiers made a correct prediction on the RofC and uses them to form an ensemble to classify the test sample. Two different schemes using KNORA were proposed:

KNORA - Eliminate (KNORA-E) Given θ_j with k neighbours from the unknown test sample $\mathbf{x}_{j,test}$, and supposing a set of classifier $\Psi \subset C = \{c_1, c_2, \ldots, c_N\}$ correctly classifies **all** the k neighbours, then every classifier $\psi \in \Psi$ will submit a vote on the sample $\mathbf{x}_{j,test}$. The label of $\mathbf{x}_{j,test}$ will be the class with the highest number of votes. If $\Psi = \varnothing$ for the k neighbours of $\mathbf{x}_{j,test}$, then the algorithm simply decreases the value of k until at least one classifier correctly classifies all neighbours.

KNORA - Union (KNORA-U) Given θ_j with k neighbours from the unknown test sample $\mathbf{x}_{j,test}$, and supposing a set of classifier $\Psi \subset C = \{c_1, c_2, \ldots, c_N\}$ correctly classifies **at least one** of the k neighbours, then each classifier $\psi \in \Psi$ will submit the same number of votes as correctly predicted neighbours. In other works, a classifier can have more than one vote if correctly classifies more than one neighbour.

Algorithm 10: KNORA-E

```
input  : $\mathbb{L}$, $\mathbb{V}$, $\mathbb{T}$ and $C$
output: $\upsilon$
/* $\upsilon$ is the array with the predicted label for all samples in $\mathbb{L}$ */
foreach $\mathbf{x}_{j,test} \in \mathbb{T}$ do
    if $\forall c_i \in C$ predict the same label $\omega_l$ for $\mathbf{x}_{j,test}$ then
        $\upsilon_j = \omega_l$ ;
    else
        $k = K$ ;
        $\Psi = \varnothing$ ;
        while $\Psi == \varnothing$ do
            Find $\theta_j$ of $\mathbf{x}_{j,test}$ using a k-NN in the validation set $\mathbb{V}$ ;
            foreach $c_i \in C$ do
                if $c_i$ correctly recognizes all samples in $\theta_j$ then
                    $\Psi = \Psi \cup c_i$ ;
            end
            if $\Psi == \varnothing$ then
                $k = k - 1$ ;
            else
                break;
            end
        end
        $\upsilon_j = MajorityVote(\mathbf{x}_{j,test}, \Psi)$ ;
    end
end
```

Algorithm 11: KNORA-U

```
input  : $\mathbb{L}$, $\mathbb{V}$, $\mathbb{T}$ and $C$
output: $\upsilon$
/* $\upsilon$ is the array with the predicted label for all samples in $\mathbb{L}$ */
foreach $\mathbf{x}_{j,test} \in \mathbb{T}$ do
    if $\forall c_i \in C$ predict the same label $\omega_l$ for $\mathbf{x}_{j,test}$ then
        $\upsilon_j = \omega_l$ ;
    else
        Find $\theta_j$ of $\mathbf{x}_{j,test}$ using a k-NN in the validation set $\mathbb{V}$ ;
        $\Psi = \varnothing$ ;
        foreach $\mathbf{x}_k \in \theta_j$ do
            foreach $c_i \in C$ do
                if $c_i(\mathbf{x}_k) == \omega_t$ then
                    $\Psi = \Psi \cup c_i$ ;
            end
        end
        $\upsilon_j = MajorityVote(\mathbf{x}_{j,test}, \Psi)$ ;
    end
end
```

A.11 Measures of Classifier Competence

This class of methods represents an ensemble dynamic selection method based on a class-independent measure of classifier competence in the feature space. All the methods are calculated with respect to the response obtained by random guessing. Therefore, these approaches gain a meaningful interpretation: competent (incompetent) classifiers are more (less) accurate than the random classifier. The competence of a classifier is determined by a two-step procedure: the first part calculates the source competence at validation points, while the second part extends these source competences to the entire feature space using a potential function model [29, 65–67].

Let $(d_1(\chi), d_2(\chi), \ldots, d_L(\chi))$ be a set of discriminant functions produced by the classifier $c_i \in C$ for a given sample described by the feature vector χ, where each $d_\omega(\chi), \Omega = \omega_1, \omega_2, \ldots, \omega_L$ represents the support given by the classifier c_i for the l-th class. Without loss of generality, $d_\omega(\chi) > 0$ and $\sum d_\omega(\chi) = 1$. Also, assume a validation set $\mathbb{V}$ is present and $\mathbf{x}_k \in \mathbb{V}$. Now using $\mathbb{V}$, we define the source competence $K(c_i \mid \mathbf{x}_k)$ of the classifier $c_i \in C$ at a point $\mathbf{x}_k \in \mathbb{V}$ as a function of class number L and the support of correct class $d_{\omega_t}(\mathbf{x}_k)$ having the following properties [29, 65–67]:

1. $K(c_i \mid \mathbf{x}_k)$ is a strictly increasing function of $d_{\omega_t}(\mathbf{x}_k)$;
2. $-1 \leq K(c_i \mid \mathbf{x}_k) \leq 1$;
3. $-1 \leq K(c_i \mid \mathbf{x}_k) < 0$ for $0 < d_{\omega_t}(\mathbf{x}_k) < \frac{1}{L} \Rightarrow$ the classifier is incompetent;
4. $K(c_i \mid \mathbf{x}_k) = -1$ for $d_{\omega_t}(\mathbf{x}_k) = 0 \Rightarrow$ the classifier is absolutely incompetent;
5. $0 < K(c_i \mid \mathbf{x}_k) \leq 1$ for $\frac{1}{L} < d_{\omega_t}(\mathbf{x}_k) < 1 \Rightarrow$ the classifier is competent;
6. $K(c_i \mid \mathbf{x}_k) = 1$ for $d_{\omega_t}(\mathbf{x}_k) = 1 \Rightarrow$ the classifier is absolutely competent;
7. $K(c_i \mid \mathbf{x}_k) = 0$ for $d_{\omega_t}(\mathbf{x}_k) = \frac{1}{L} \Rightarrow$ the classifier is neutral

The following source competence functions were proposed by Antosik & Kurzynski [66], Woloszynski & Kurzynski [29, 65] and Woloszynski *et al.* [67]:

DES - Minimal Difference (DES-MD) [66] First, this function calculates the difference between the discriminant function obtained by the classifier c_i for the correct class ($d_{\omega_t}(\mathbf{x}_k)$) and those obtained by c_i for each of the other classes ($d_{\omega}(\mathbf{x}_k)$ with $\omega \neq \omega_t$). The difference with the minimal value is selected as the source competence of the classifier c_i. If c_i correctly classifies $\mathbf{x}_k \in \mathbb{V}$, then $K(c_i \mid \mathbf{x}_k) > 0$ and c_i is considered competent. If c_i makes an error, then $K(c_i \mid \mathbf{x}_k) < 0$ and c_i is incompetent.

$$K(c_i \mid \mathbf{x}_k) = \min_{\substack{\omega \in \Omega \\ \omega \neq \omega_t}} [d_{\omega_t}(\mathbf{x}_k) - d_{\omega}(\mathbf{x}_k)] \tag{A.13}$$

DES - Exponential (DES-EXP) [65] In this function, the source competence $K(c_i \mid \mathbf{x}_k)$ depends on the number of classes in the classification problem. This dependency can be observed on Figure A.7, where by increasing the number of classes, the chances of having a higher source competence is increased, even with a low support given by the classifier. Therefore, a low number of classes is desired for this function to properly work.

$$K(c_i \mid x_k) = 1 - 2^{1 - \dfrac{(L-1)d_{\omega_t}(\mathbf{x}_k)}{1 - d_{\omega_t}(\mathbf{x}_k)}} \tag{A.14}$$

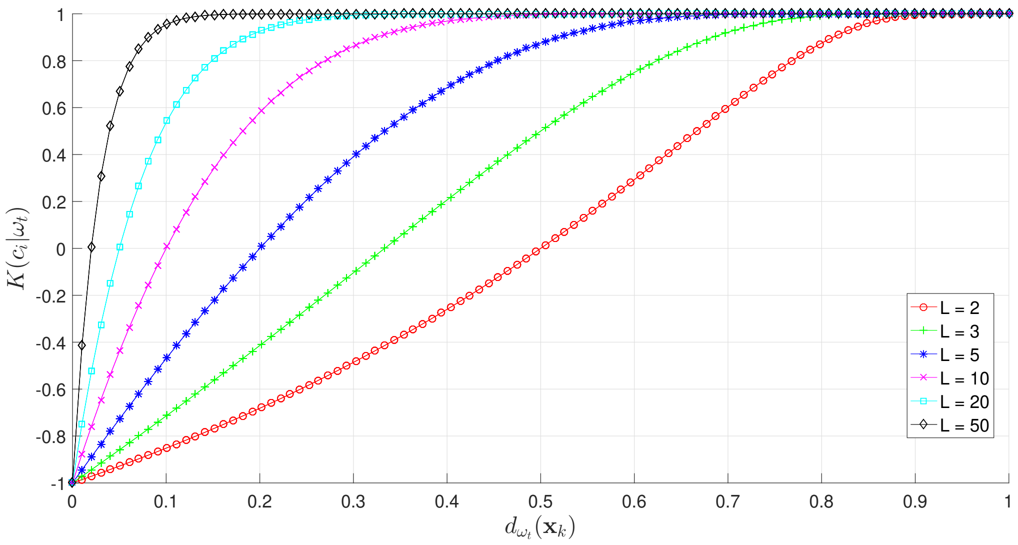


Figure A.7: Exponential function number of classes dependency. The source competence of classifier c_i plotted against the support predicted by c_i for the correct class $c_i(\mathbf{x}_k) = \omega_t$ (Equation A.14) for different number of classes (L = 2, 3, 5, 10, 20, 50)

DES - Randomised Reference Classifier (DES-RRC) [29] This method decides whether or not a base classifier performs better than a random guessing. The source competence $K(c_i \mid \mathbf{x}_k)$ is estimated based on the randomised reference classifier with a beta probability distribution proposed by Woloszynski & Kurzynski [29]. A MATLAB code *ccprmod.m* is freely available for download with the proposed function [1].

DES - Kullback-Leibler Divergence (DES-KL) [67] This method measures the source competence of a classifier from the information theory perspective. The Kullback-Leibler divergence is computed between the uniform distribution $RC = [\frac{1}{L}, \frac{1}{L}, \ldots, \frac{1}{L}]$ and the vector of discriminant functions $(d_1(\chi), d_2(\chi), \ldots, d_L(\chi))$ produced by the classifier $c_i \in C$. Therefore, this divergence measures how "close" the discriminant functions are to the random guessing.

[1]The *ccprmod.m* code is available for download at MATLAB File Exchange: https://www.mathworks.com/matlabcentral/fileexchange/28391-a-probabilistic-model-of-classifier-competence [29]

$$K(c_i \mid \mathbf{x}_k) = (2 * I_{\{c_i(\mathbf{x}_k)=\omega_t\}} - 1) + \sum_{l=1}^{L} d_\omega(\mathbf{x}_k) \log \left(\frac{d_\omega(\mathbf{x}_k)}{RC(l)} \right) \tag{A.15}$$

$I_{\{c_i(\mathbf{x}_k)=l_k\}}$ is the indicator of the event $c_i(\mathbf{x}_k) = l_k$, i.e., the classifier c_i made a correct prediction of the class of the validation sample $\mathbf{x}_k$.

Since the Kullback-Leibler divergence is non-negative, the term $(2 * I_{\{c_i(\mathbf{x}_k)=l_k\}} - 1)$ gives the sign of the source competence $K(c_i \mid \mathbf{x}_k)$. In other words, the source competence is positive (negative) if the classifier correctly (incorrectly) predicts the label of the validation sample $\mathbf{x}_k$.

After calculating the source competence $K(c_i \mid \mathbf{x}_k)$ at validation points $\mathbf{x}_k \in \mathbb{V}$, the following step is to extend the competence values to the entire feature space using a normalized Gaussian potential function model to reduce the influence of each point in the validation set based on its Euclidean distance to $\mathbf{x}_{j,test}$:

$$\delta_{i,j} = \frac{\sum_{\mathbf{x}_k \in \mathbb{V}} K(c_i \mid \mathbf{x}_k) e^{-d(\mathbf{x}_{j,test}, \mathbf{x}_k)^2}}{\sum_{\mathbf{x}_k \in \mathbb{V}} e^{-d(\mathbf{x}_{j,test}, \mathbf{x}_k)^2}} \tag{A.16}$$

where $d(\mathbf{x}_{j,test}, \mathbf{x}_k)$ is the Euclidean distance between the unknown test sample $\mathbf{x}_{j,test}$ and the validation sample $\mathbf{x}_k$. All classifiers that achieve a greater competence than the probability of a random classifier is selected to compose the ensemble to predict the label of the unknown test sample $\mathbf{x}_{j,test}$ using a majority voting scheme [29, 65–67].

A.12 DES - Performance (DES-P)

The DES-P method was proposed by Woloszynski *et al.* [67] and is similar to the OLA algorithm. First, the performance of each classifier $c_i \in C$ is computed in θ_j by calculating the percentage of neighbours c_i correctly classified. Then, the competence $\delta_{i,j}$ of c_i is obtained by subtracting the performance of a random classifier $\frac{1}{L}$ from the estimated performance of c_i.

$$\delta_{i,j} = \frac{N_i}{K} - \frac{1}{L} \tag{A.17}$$

Algorithm 12: DES-MD / DES-EXP / DES-RRC / DES-KL

```
input  : $\mathbb{L}$, $\mathbb{V}$, $\mathbb{T}$ and $C$
output: $\upsilon$
/* $\upsilon$ is the array with the predicted label for all samples in $\mathbb{L}$ */
1  foreach $\mathbf{x}_{j,test} \in \mathbb{T}$ do
2      if $\forall c_i \in C$ predict the same label $\omega_l$ for $\mathbf{x}_{j,test}$ then
3          $\upsilon_j = \omega_l$ ;
4      else
5          $\Psi = \varnothing$ ;
6          $\Psi' = \varnothing$ ;
7          foreach $c_i \in C$ do
8              Calculate $\delta_{i,j}$ using equation A.16 with one of the four types of source
                 competence measures: minimal difference (equation A.13),
                 exponential (equation A.14), randomised reference classifier ,
                 Kullback-Leibler divergence (equation A.15) ;
9              if $\delta_{i,j} \geq \frac{1}{L}$ then
10                 $\Psi = \Psi \cup c_i$ ;
11             if $\delta_{i,j} > 0$ then
12                 $\Psi' = \Psi' \cup c_i$ ;
13         end
14         if $\Psi \neq \varnothing$ then
15             $\upsilon_j = MajorityVote(\mathbf{x}_{j,test}, \Psi)$ ;
16         else
17             $\upsilon_j = MajorityVote(\mathbf{x}_{j,test}, \Psi')$ ;
18         end
19     end
20 end
```

where N_i is the number of neighbouring samples of $\mathbf{x}_{j,test}$ that are correctly classified by c_i and K is the size of the RofC.

The classifiers with a positive competence $\delta_{i,j}$ will form the ensemble to predict the label of $\mathbf{x}_{j,test}$ using a majority voting scheme.

A.13 k-Nearest Output Profiles (KNOP)

Proposed by Cavalin *et al.* [68], the KNOP method is inspired on the KNORA algorithms and transforms all samples in the validation set $\mathbb{V}$ and the unknown test sample $\mathbf{x}_{j,test}$ into output profiles to create θ_j. The k most similar validation samples according to equation A.6 will be selected to form θ_j. The main differences of this method when

Algorithm 13: DES-P

```
input  : $\mathbb{L}$, $\mathbb{V}$, $\mathbb{T}$ and $C$
output: $\upsilon$
/* $\upsilon$ is the array with the predicted label for all samples in $\mathbb{L}$ */
1  foreach $\mathbf{x}_{j,test} \in \mathbb{T}$ do
2      if $\forall c_i \in C$ predict the same label $\omega_l$ for $\mathbf{x}_{j,test}$ then
3          $\upsilon_j = \omega_l$ ;
4      else
5          Find $\theta_j$ of $\mathbf{x}_{j,test}$ using a k-NN in the validation set $\mathbb{V}$ ;
6          $\Psi = \varnothing$ ;
7          foreach $c_i \in C$ do
8              Calculate $\delta_{i,j}$ using equation A.17 ;
9              if $\delta_{i,j} > 0$ then
10                 $\Psi = \Psi \cup c_i$ ;
11         end
12         $\upsilon_j = MajorityVote(\mathbf{x}_{j,test}, \Psi)$ ;
13     end
14 end
```

compared with the MCB method are:

Algorithm 14: KNOP-U

```
input  : $\mathbb{L}$, $\mathbb{V}$, $\mathbb{T}$ and $C$
output: $\upsilon$
/* $\upsilon$ is the array with the predicted label for all samples in $\mathbb{L}$ */
1  Transform all the sample in $\mathbb{V}$ into output profiles $\tilde{\mathbf{x}}_v$ ;
2  foreach $\mathbf{x}_{j,test} \in \mathbb{T}$ do
3      if $\forall c_i \in C$ predict the same label $\omega_l$ for $\mathbf{x}_{j,test}$ then
4          $\upsilon_j = \omega_l$ ;
5      else
6          Compute the output profile $\tilde{\mathbf{x}}_{j,test}$ of $\mathbf{x}_{j,test}$ ;
7          Find the $k$ most similar $\tilde{\mathbf{x}}_v$ to $\tilde{\mathbf{x}}_{j,test}$ to compute $\theta_j$ ;
8          $\Psi = \varnothing$ ;
9          foreach $\mathbf{x}_k \in \theta_j$ do
10             foreach $c_i \in C$ do
11                 if $c_i(\mathbf{x}_k) == \omega_t$ then
12                     $\Psi = \Psi \cup c_i$ ;
13             end
14         end
15         $\upsilon_j = MajorityVote(\mathbf{x}_{j,test}, \Psi)$ ;
16     end
17 end
```

1. The MCB method first uses a k-NN to compute θ_j and then selects the most similar

output profiles using the equation A.6; while the KNOP method will select the k most similar profiles using the entire validation set $\mathbb{V}$.

2. The MCB method selects a single classifier while the KNOP algorithm selects an ensemble of classifiers.

Similarly to the KNORA algorithms there are two possible KNOP methods: KNOP - Eliminate (KNOP-E) and KNOP - Union (KNOP-U); depending on how the classifiers will be selected to form the ensemble.

Algorithm 15: KNOP-E

```
input  : $\mathbb{L}$, $\mathbb{V}$, $\mathbb{T}$ and $C$
output: $\upsilon$
/* $\upsilon$ is the array with the predicted label for all samples in $\mathbb{L}$ */
Transform all the sample in $\mathbb{V}$ into output profiles $\tilde{\mathbf{x}}_k$ ;
foreach $\mathbf{x}_{j,test} \in \mathbb{T}$ do
    if $\forall c_i \in C$ predict the same label $\omega_l$ for $\mathbf{x}_{j,test}$ then
        $\upsilon_j = \omega_l$ ;
    else
        $k = K$ ;
        $\Psi = \varnothing$ ;
        while $\Psi == \varnothing$ do
            Compute the output profile $\tilde{\mathbf{x}}_{j,test}$ of $\mathbf{x}_{j,test}$ ;
            Find the $k$ most similar $\tilde{\mathbf{x}}_k$ to $\tilde{\mathbf{x}}_{j,test}$ to compute $\theta_j$ ;
            foreach $c_i \in C$ do
                if $c_i$ correctly recognizes all samples in $\theta_j$ then
                    $\Psi = \Psi \cup c_i$ ;
            end
            if $\Psi == \varnothing$ then
                $k = k - 1$ ;
            else
                break;
            end
        end
        $\upsilon_j = MajorityVote(\mathbf{x}_{j,test}, \Psi)$ ;
    end
end
```

A.14 Meta-learning - DES (Meta-DES)

Proposed by Cruz *et al.* [30], the Meta-DES method is a DES framework that uses meta-learning. Most DS techniques use either one or two criteria to select a classifier or an ensemble of classifiers from a RofC θ_j surrounding an unknown test sample $\mathbf{x}_{j,test}$. However, this criterion can be very error-prone and, according to the “No Free Lunch” theorem, no algorithm is better in all possible class of problems. Therefore, to overcome these issues, Cruz *et al.* [30] proposed a method with multiple criteria (meta-features) to measure the competence of the base classifiers $c_i \in C$ to form an ensemble. The authors propose five different meta-features, each one capturing a different behaviour of c_i:

f_1 **- Neighbour's hard classification** First, a vector f_1 with K elements is created. Then, from each sample $\mathbf{x}_k \in \theta_j$, if c_i correctly classifies $\mathbf{x}_k$, the k-th position of f_1 is set to 1, otherwise is set to 0.

f_2 **- Posterior probability** First, a vector f_2 with K elements is created. Then, from each sample $\mathbf{x}_k \in \theta_j$, the posterior probability $P(\omega_t|\mathbf{x}_k \subset \theta_j, c_i)$ is computed and inserted into the k-th position of f_2.

f_3 **- Overall local accuracy** The accuracy of c_i over θ_j is computed using the equation A.1.

f_4 **- Output profiles classification** Similar to the KNOP method, first a vector f_4 with K_p elements is created. Then, all samples in the validation set $\mathbb{V}$ are transformed into output profiles $\tilde{\mathbf{x}}_k$ and the K_p most similar $\tilde{\mathbf{x}}_k$ to $\tilde{\mathbf{x}}_{j,test}$ are selected to form $\tilde{\theta}_j$. From each sample $\tilde{\mathbf{x}}_{k_p} \in \tilde{\theta}_j$, if c_i correctly classifies $\tilde{\mathbf{x}}_{k_p}$, the k-th position of f_4 is set to 1, otherwise is set to 0.

f_5 **- Classifier's confidence** The posterior probability of the class w_l predict by c_i for $\mathbf{x}_k$, i.e., $P(c_i(\mathbf{x}_k) = \omega_l|\mathbf{x}_k \subset \theta_j, c_i)$

In consequence, if one of this meta-features fail (have a low confidence result), the system can still achieve a good performance. These meta-feature will then be used to train a meta-classifier that decides whether or not the classifier c_i is competent to predict the label of $\mathbf{x}_{j,test}$. Cruz *et al.* [115] evaluated four types of meta-classifiers: Multiple Layer Perceptron Neural Network (MLP), Support Vector Machine (SVM), Random Forest classifier (RF) and Naive Bayes classifier (NB). Experimental results demonstrated that the performance of this classifiers were statistically equivalent; hence, since NB is a simpler classifier to train due to the lack of hyper-parameters, this classifier was chosen as the meta-classifier.

The Meta-DES method is divided into three phases:

1. **Overproduction:** a pool of classifiers (C) is generated using a bagging method, which builds a diverse ensemble of classifiers by randomly selecting different subsets of the training set $\mathbb{T}$. Each subset of $\mathbb{T}$ is used to train one classifier $c_i \in C$.

2. **Meta-training:** the five sets of meta-features are extracted from the set $\mathbb{T}_\lambda$ and used to train the meta classifier λ.

3. **Generalization:** the meta-features are extracted from the unknown test sample $\mathbf{x}_{j,test}$ and passed as an input to λ, which decides if the classifier c_i is competent or not to predict the label of $\mathbf{x}_{j,test}$. If it is competent, c_i will form the ensemble that will predict the label of $\mathbf{x}_{j,test}$ using a majority vote scheme.

The meta-training and generalization phases are formalized in Algorithm 16. The proposed framework differs from all the other dynamic selection algorithms in two aspects: (1) it uses a multiple selection criterion to determine a classifier's competence; (2) the rule to select a classifier is learned by a meta-classifier using these multiple criteria.

A.15 Dynamic Selection on Complexity (DSOC)

Proposed by Brun *et al.* [69], the DSOC method combines the accuracy of a classifier $c_i \in C$ on the RofC θ_j with the complexity of the data to select a single classifier from the pool C. The complexity of the data is measured by complexity features, which analyse the level of difficulty of a classification problem. These features analyse not only the number of instances, classes and features, but also how overlapping are two classes, how are their boarders and the spatial distribution of each class. Three complexity measures were proposed by Brun *et al.* [69] to form the DSOC method:

Fisher's Discriminant Ratio $F1$ describes how separable are two classes according to each feature.

$$F1 = \frac{(\mu_1 - \mu_2)^2}{{\sigma_1}^2 - {\sigma_2}^2} \tag{A.18}$$

where μ_1,μ_2,σ_1 and σ_2 represent the mean value and the standard deviation of for classes 1 and 2 for each feature in the feature space. The final value of $F1$ is the highest among all features. The higher $F1$ is, the larger the separation between two classes is, because $F1$ can be interpreted as the distance between the centre of two classes.

Ratio of intra/inter class nearest neighbour distance $N2$ analyses the existence and form of the border between two classes to determine how separable they are. Therefore, $N2$ calculates the sum of distance between the sample x_i and its nearest neighbour in the same class and divides by the sum of distance between the same sample x_i and its nearest neighbour outside x_i's class. The smaller the $N2$, more separable the classes are.

$$N2 = \frac{\sum_{i=1}^{n} \rho(N_1^{=}(x_i), x_i)}{\sum_{i=1}^{n} \rho(N_1^{\neq}(x_i), x_i)} \tag{A.19}$$

where $\rho(N_1^{=}(x_i), x_i)$ represents the distance between x_i and its nearest neighbour in the same class; $\rho(N_1^{/} = (x_i), x_i)$ represents the distance between x_i and its nearest neighbour in a different class; and n is the number of instances.

Non-linearity of the 1-Nearest Neighbour classifier $N4$ describes the error rate of a training set and a test set using a 1-Nearest Neighbour (1-NN) classifier. The test set $\mathcal{T}$ is created by linearly interpolating ($\bar{x} = \alpha x_k + (1 - \alpha)x_l$) elements of randomly chosen pairs (x_k and x_l) within the same class in the training set, where $\alpha \in [0,1]$. The smaller $N4$ is, the easier is the problem.

$$N4 = 1NN_{error}(\mathcal{T}) = \frac{\mathcal{N}}{n} \tag{A.20}$$

where n is the number of instances in $\mathcal{T}$ and $\mathcal{N}$ is the number of errors 1NN made, i,e, the number of times 1NN selects the incorrect class for $\bar{x} \in \mathcal{T}$.

These complexity features are computed over each subset of data (DS_i, where $i = 1, \ldots, N$ and N is equal to the total number of classifiers in the pool) generated by bagging, creating a M-complexity signature array (sig_{DS_i}). Also, for each DS_i the centroid of each class $\alpha_{i,j}$ is computed. Then, for each $\mathbf{x}_{j,test} \in \mathbb{L}$, θ_j is computed to find another M-complexity signature array (sig_{θ_j}). Finally, using sig_{DS_i}, $\alpha_{i,j}$, sig_{θ_j} and θ_j three features are extracted for each classifier, and their combination (Equation A.21) will decide which classifier is more competent to predict the label of $\mathbf{x}_{j,test}$, i.e., the classifier with the highest $\delta_{i,j}$ will be chosen.

f_{1_i} **- Similarity in terms of complexity** Consider sig_{DS_i} as the M-complexity signature array for each DS_i. Also, consider sig_{θ_j} as the M-complexity signature array of θ_j. The similarity between sig_{θ_j} and each sig_{DS_i} is computed using the Euclidean distance.

f_{2_i} **- Distance from the predicted class** First, consider that ω_l is the class predicted by c_i for $\mathbf{x}_{j,test}$. Also, let $\alpha_{i,j}$ be the centroid of class ω_l in each DS_i. Then, we compute the Euclidean distance between $\alpha_{i,j}$ and $\mathbf{x}_{j,test}$, creating an N-dimensional array.

f_{3_i} **- Local class accuracy** exploits the information that $c_i(\mathbf{x}_{j,test}) = \omega_l$, determining, therefore, the percentage of neighbouring sample correctly assigned to ω_l by c_i. This can be calculated using equation A.2.

$$\delta_{i,j} = (1 - f'_{1_i}) + (1 - f'_{2_i}) + f_{3_i} \tag{A.21}$$

where f'_{1_i} and f'_{2_i} correspond to the normalized metrics of f_{1_i} and f_{2_i}, calculated using the Min-Max scaling scheme:

$$f'_n = \frac{f_n - \min(f_n)}{\max(f_n) - \min(f_n)} \tag{A.22}$$

Algorithm 16: Meta-DES

```
input  : $\mathbb{L}$, $\mathbb{V}$, $\mathbb{T}$, $\mathbb{T}_\lambda$ and $C$
output: $\upsilon$
/* $\upsilon$ is the array with the predicted label for all samples in $\mathbb{L}$ */
1  $\mathbb{T}^*_\lambda = \varnothing$;
2  foreach $\mathbf{x}_{j,train} \in \mathbb{T}_\lambda$ do
3      Compute the accuracy $\rho$ of the pool over $\mathbf{x}_{j,train}$;
       /* $\rho$ is the number of correct predictions divided by the total number of
          classifiers */
4      if $\rho < threshold$ then
5          Find $\theta_{j,\lambda}$ using $\mathbb{T}_\lambda$ ;
6          Transform all the samples $\mathbf{x}_{j,train} \in \mathbb{T}_\lambda$ into output profiles $\tilde{\mathbf{x}}_{j,train} \in \tilde{\mathbb{T}}_\lambda$ ;
7          Find the $K_p$ most similar output profiles of $\tilde{\mathbb{T}}_\lambda$ to $\tilde{\mathbf{x}}_{j,train}$, excluding
            $\tilde{\mathbf{x}}_{j,train}$, to form the region $\phi_{j,lambda}$ ;
8          foreach $c_i \in C$ do
9              Compute $f_1$, $f_2$, $f_3$, $f_4$ and $f_5$ using $\phi_{j,lambda}$ and $\theta_{j,\lambda}$;
10             $v_{i,j} = \{f_1 \cup f_2 \cup f_3 \cup f_4 \cup f_5\}$ ;
11             if $c_i$ correctly classifies $\mathbf{x}_{j,train}$ then
12                 $\alpha_{i,j} = 1$
13             else
14                 $\alpha_{i,j} = 0$
15             end
16             $\mathbb{T}^*_\lambda = \mathbb{T}^*_\lambda \cup \{v_{i,j}\}$
17         end
18 end
19 Train the meta-classifier $\lambda$ with $\mathbb{T}^*_\lambda$ and $\alpha$
20 foreach $\mathbf{x}_{j,test} \in \mathbb{T}$ do
21     if $\forall c_i \in C$ predict the same label $\omega_l$ for $\mathbf{x}_{j,test}$ then
22         $\upsilon_j = \omega_l$ ;
23     else
24         $\Psi = \varnothing$ ;
25         Find $\theta_j$ of $\mathbf{x}_{j,test}$ using the validation set $\mathbb{V}$ ;
26         Transform $\mathbf{x}_{j,test}$ into $\tilde{\mathbf{x}}_{j,test}$ ;
27         Transform all the samples $\mathbf{x}_k \in \mathbb{V}$ into output profiles $\tilde{\mathbf{x}}_k \in \tilde{\mathbb{V}}$ ;
28         Find the $K_p$ most similar output profiles of $\tilde{\mathbb{V}}$ to $\tilde{\mathbf{x}}_{j,test}$, to form the region
            $\phi_j$ ;
29         foreach $c_i \in C$ do
30             Compute $f_1$, $f_2$, $f_3$, $f_4$ and $f_5$ using $\phi_j$ and $\theta_j$;
31             $v_{i,j} = \{f_1 \cup f_2 \cup f_3 \cup f_4 \cup f_5\}$;
32             if $\lambda(v_{i,j}) == 1$ then
33                 $\Psi = \Psi \cup \{c_i\}$;
34         end
35         $\upsilon_j = MajorityVote(\mathbf{x}_{j,test}, \Psi)$
36     end
37 end
```

Algorithm 17: DSOC

```
input  : $\mathbb{L}$, $\mathbb{V}$, $\mathbb{T}$ and $C$
output: $\upsilon$
/* $\upsilon$ is the array with the predicted label for all samples in $\mathbb{L}$ */
1  foreach $c_i \in C$ do
2      Compute $sig_{DS_i}$, from data subset $DS_i$ ;
3  end
4  foreach $\mathbf{x}_{j,test} \in \mathbb{T}$ do
5      if $\forall c_i \in C$ predict the same label $\omega_l$ for $\mathbf{x}_{j,test}$ then
6          $\upsilon_j = \omega_l$ ;
7      else
8          Find $\theta_j$ of $\mathbf{x}_{j,test}$ using a k-NN in $\mathbb{V}$ ;
9          Compute $sig_{\theta_j}$ of $\theta_j$ ;
10         foreach $c_i \in C$ do
11             foreach $DS_i$ do
12                 Compute $f_{1_i}$ and $f_{2_i}$ ;
13             end
14             Compute $f_{3_i}$ ;
15             Normalize $f_{1_i}$ and $f_{2_i}$ ;
16             $\delta_{i,j} = (1 - f'_{1_i}) + (1 - f'_{2_i}) + f_{3_i}$ ;
17         end
18         $C^* = argmax(\delta_{i,j})$ ;
19         $\upsilon_j = C^*(\mathbf{x}_{j,test})$ ;
20     end
21 end
```

Appendix B

Statistical Analysis

***

Over the last years, the machine learning community has used different statistical methods to validate their results. Pairwise and non-pairwise t-test, averages and counts of wins are some of the most common statistical tests in the literature to answer one of the most import questions in machine learning: which classifier yields an improved score when comparing with other classifiers or state-of-art methods? [98, 116]. When the differences between classifiers are very clear, i.e., when one classifier has the best score in all datasets studied, the direct comparison may be enough. However, in most situations, this is not the case. Hence, statistical methods must be employed to determine the relevance of these differences [116].

Statistical methods are employed to give answers to the above question and they provide a more precise technique to determine if the differences are random or real. In 2006, Demsar [98] evaluated the usage of several statistical tests in different papers and concluded that:

1. Comparisons of two classifiers over a single dataset have been scrutinised by the

machine learning community and there are well established methods in the literature and;

2. The choice of statistical methods to compare multiple classifiers over multiple datasets (an increasing common situation) was not well established and researchers were unsure about which tests were appropriate for these cases.

Regarding the second conclusion, two situations are possible: (1) the comparison of two classifiers over multiple datasets; and (2) the comparison of multiple classifier over multiple datasets. The following sections (sections B.1 and B.2) discusses in details the most common methods used in literature for these two cases, as well as their misuses, limitations, advantages and disadvantages. Section B.3 discuss the *Wilcoxon rank sum test* to compare DCS and DES methods.

B.1 Comparison of two classifiers

For this class of problem, Demsar [98] analysed four different statistical methods: averaging, paired t-test, sign test and Wilcoxon signed-rank test.

B.1.1 Averaging over datasets

In the words of Webb [117] “it is debatable whether error rates in different domains are commensurable, and hence whether averaging error rates across domains is very meaningful”. For this reason, Demsar [98] states that if the datasets are comparable (set of related problems, for example, different medical databases of a certain disease), averaging their results are meaningful, otherwise their averages are meaningless. Also, average is susceptible to outliers, i.e., if one classifier performs excellent in a single dataset, the overall bad performance on the other datasets might be underestimated.

B.1.2 Paired t-test

According to Demsar [98], a common way in the literature to determine if the difference between two classifiers over multiple datasets are significantly different from zero is to compute a *paired t-test*, which has a Student distribution.

In the context of comparing classifiers over multiple datasets, the *paired t-test* suffers from three weaknesses:

1. **Commensurability**. For the same reasons stated in the averaging case, if the datasets are not comparable, the comparison of the differences between classifiers is also meaningless [98].

2. **Assumption of normal distribution**. The *paired t-test* requires that the differences between two random variables compared are distributed normally. However, this is not usually the case, since the nature of the problem does not suggest a normal distribution and, also, the number of datasets studied is much less than 30 [98, 116]. Demsar [98] stated that "for using the t-test we need normal distributions because we have small samples, but the small samples also prohibit from checking the distribution shape".

3. **Outliers**. The *paired t-test* is also affected by outliers which can decrease the test's power by increasing the estimated standard error.

B.1.3 Sign test and count of wins, losses and ties

According to Demsar [98], a popular way to compare the performance of two classifiers over multiple datasets is to count the number of times a classifier performs better, worst or equivalent to another classifier. Also, by using these counts in inferential statistics with a binomial test (*sign test*). If two classifiers are equivalent, as assumed under the null-hypothesis, each one should win approximately half on the total number of datasets. Hence, the number of wins is distributed according to a binomial distribution.

Table B.1: Critical values of the two-tailed sign test (adapted from Mishra & Osman [118])

# datasets	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$\omega_{0.01}$	5	6	7	7	8	9	9	10	10	11	12	12	13	13	14	15	15	16	16	17	18
$\omega_{0.05}$	5	6	7	8	8	9	10	10	11	12	12	13	13	14	15	15	16	17	17	18	18
$\omega_{0.10}$	5	6	7	8	9	10	11	11	12	13	13	14	15	15	16	17	17	18	19	19	20

The sign test does not assume commensurabitility and normal distribution. In consequence, it can be used in any set of data. However, according to the critical values presented in table B.1, one classifier can be deemed better, if it performs almost always better than another classifier [98]

B.1.4 Wilcoxon signed-rank test

The *Wilcoxon signed-rank test* is a non-parametric test that is an alternative to the *paired t-test*. Demsar [98] recommends this test for comparing two classifier over multiple datasets because it does not assume normal distributions and the outliers have less influence.

This test ignores the signs and absolute values of classifier's difference by ranking them. The smallest difference gets rank 1 and in case of ties, the average ranks are used. Let $A_{c_1}^{(i)}$ and $A_{c_2}^{(i)}$ be the score of two classifiers c_1 and c_2 obtained on the i-th of N datasets and $d_i = |A_{c_1}^{(i)} - A_{c_2}^{(i)}|$ be the absolute value of the difference between the scores. The test statistics of the rank of the differences d_i is:

$$T = \min(R^+, R^-) \tag{B.1}$$

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

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{B.3}$$

A table with the exact critical values of the test statistic can be found on most statistical text book [119]. For a larger number of datasets (over 30), the statistics:

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

is distributed normally [98, 116, 120].

B.2 Comparison of multiple classifiers

When considering the comparison of multiple classifiers over multiple datasets, there are three common approaches according to Demsar [98]:

B.2.1 Pairwise

Many researches according to Demsar [98] use the methods mentioned on section B.1 by conducting all the pairwise results from a set of classifiers and report results like "classifier A was found significantly better than B and C, classifier D was significantly better than classifiers A and E, while there was no significant differences between the other pairs". Demsar [98] stated that when the number of pairwise comparison increases the likelihood of randomly rejecting a certain proportion of null hypothesis also increases, therefore this approach is statistically meaningless to compare multiple classifiers.

B.2.2 ANOVA

The *repeated measures ANOVA* is one of the most common statistical methods to compare the differences between more than two related samples. Although this is common statistical method, the required assumptions cannot be guaranteed when dealing with machine learning algorithms. First, ANOVA assumes that the related samples are drawn from a normal distribution. Second, and most important, assumption of *repeated measures ANOVA* is sphericity, which requires the variance of the differences between all combinations to be equal. Therefore, this method does not seem to be the ideal one to handle the comparison of multiple classifiers over multiple datasets [98, 120, 121].

B.2.3 Friedman test and Iman-Davenport extension

The *Friedman test* is a non-parametric test similar to the *repeated measures ANOVA*. If the assumptions of ANOVA are violated, the Friedman test can be more powerful since it does not assumes normal distribution and sphericity. It compares k classifiers over N datasets, by ranking the classifiers for each dataset separately (the best classifiers gets rank 1 and in case of ties, average ranks are assigned). The null hypothesis state that all classifiers are equivalent and so their ranks are equal, hence the Friedman statistic is:

$$\chi_F^2 = \frac{12N}{k(k+1)}\left(\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right) \tag{B.5}$$

is distributed according to χ_F^2 with $(k-1)$ degrees of freedom; where $R_j = \frac{1}{N}\sum_i r_i^j$ is the average rank of classifiers and r_i^j is the rank of the j-th classifier of the i-th dataset [98, 116, 120, 121].

In 1980, Iman & Davenport [97] showed that χ_F^2 has a conservative behaviour and, in consequence, proposed a better alternative:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2} \tag{B.6}$$

which is distributed according to a F distribution with $k-1$ and $(k-1)(n-1)$ degrees of freedom [1]. If the null hypothesis of equivalence of classifiers is rejected, a *post hoc* can be used to identify which classifiers performed different.

Demsar [98], suggested the *Nemenyi test* to compare all classifiers against each other. The perform of two classifiers is significantly different if their average ranking differs by more than a critical distance CD:

$$CD = q_\alpha\sqrt{\frac{k(k+1)}{6N}} \tag{B.7}$$

where q_α are based on the Studentized range statistic divided by $\sqrt{2}$ (Table B.2)

Table B.2: Critical values of the two-tailed Nemenyi test (adapated from Carvalho & Zanchettin [122])

# classifiers	2	3	4	5	6	7	8	9	10	11	12	13
$\omega_{0.01}$	2.576	2.913	3.113	3.255	3.364	3.452	3.526	3.590	3.646	3.696	3.741	3.781
$\omega_{0.05}$	1.960	2.344	2.569	2.728	2.850	2.948	3.031	3.102	3.164	3.219	3.268	3.313
$\omega_{0.10}$	1.645	2.052	2.291	2.460	2.589	2.693	2.780	2.855	2.920	2.978	3.030	3.077
# classifiers	14	15	16	17	18	19	20	21	22	23	24	25
$\omega_{0.01}$	3.818	3.853	3.884	3.914	3.941	3.967	3.992	3.015	4.037	4.057	4.077	4.096
$\omega_{0.05}$	3.354	3.391	3.426	3.458	3.489	3.517	3.544	3.569	3.593	3.616	3.637	3.658
$\omega_{0.10}$	3.120	3.159	3.196	3.230	3.261	3.291	3.319	3.346	3.371	3.394	3.471	3.439

[1]Critical values for the F distribution can be calculated using https://www.danielsoper.com/statcalc/calculator.aspx?id=4

B.3 Wilcoxon rank sum test

The *Wilcoxon rank-sum test* or *Mann Whitney U test* are essentially identical non-parametric tests which are based on the order in which the observations from two samples fall. The main difference between them is the test statistic. Suppose that we have observations from two populations A and B containing N_A and N_B observations each. The null hypothesis of the test is that the distributions of observations in the populations A and B are equivalent. It is worth noticing that this test does not assume the populations to be normally distributed. The alternative hypothesis, the two-tailed alternative, states that the distributions are different, regardless the direction of the shift [123].

The *Wilcoxon rank-sum test* is based on the rank of the combined observations of populations A and B, each observation is than ranked: the smallest value has rank 1 and, in case of ties, the average rank is used. Therefore, the *Wilcoxon rank-sum test* is valid for data with any distribution [123].

For a large number of samples, the distribution of a rank sum W_A with observed rank w_A can be approximated to a normal distribution with mean μ_A and standard deviation σ_A, where

$$\mu_A = \frac{N_A(N_A + N_B + 1)}{2} \tag{B.8}$$

$$\sigma_A = \sqrt{\frac{N_A N_B(N_A + N_B + 1)}{12}} \tag{B.9}$$

In other words, the probability of $W_A \geq w_A$ can be approximated to the probability of $Z \geq z$, where

$$z = \frac{w_A - \mu_A}{\sigma_A} \tag{B.10}$$

and Z is a normal distribution with zero mean and unit standard deviation [123].

According to Tomczak & Tomczak [124], it is widely recommended to report the effect size value of inferential test to not rely only on p-values. Effect size is a simple way to quantify the differences between two populations that reflects the strength of the relationship between the two populations. The effect size estimates, therefore, the importance of this relationship, and allows the results to be properly compared, without depending on the size of the populations.

The effect size value for the *Wilcoxon rank sum test* is given by the correlation coefficient r [124]

$$r = \frac{z}{\sqrt{N_A + N_B}} \tag{B.11}$$

A rough estimate is that $r = 0.5$ represents a large effect size, $r = 0.3$ a medium effect size and $r = 0.1$ a small effect size [125].

Appendix C

Hubness Analysis

***

C.1 Allaml

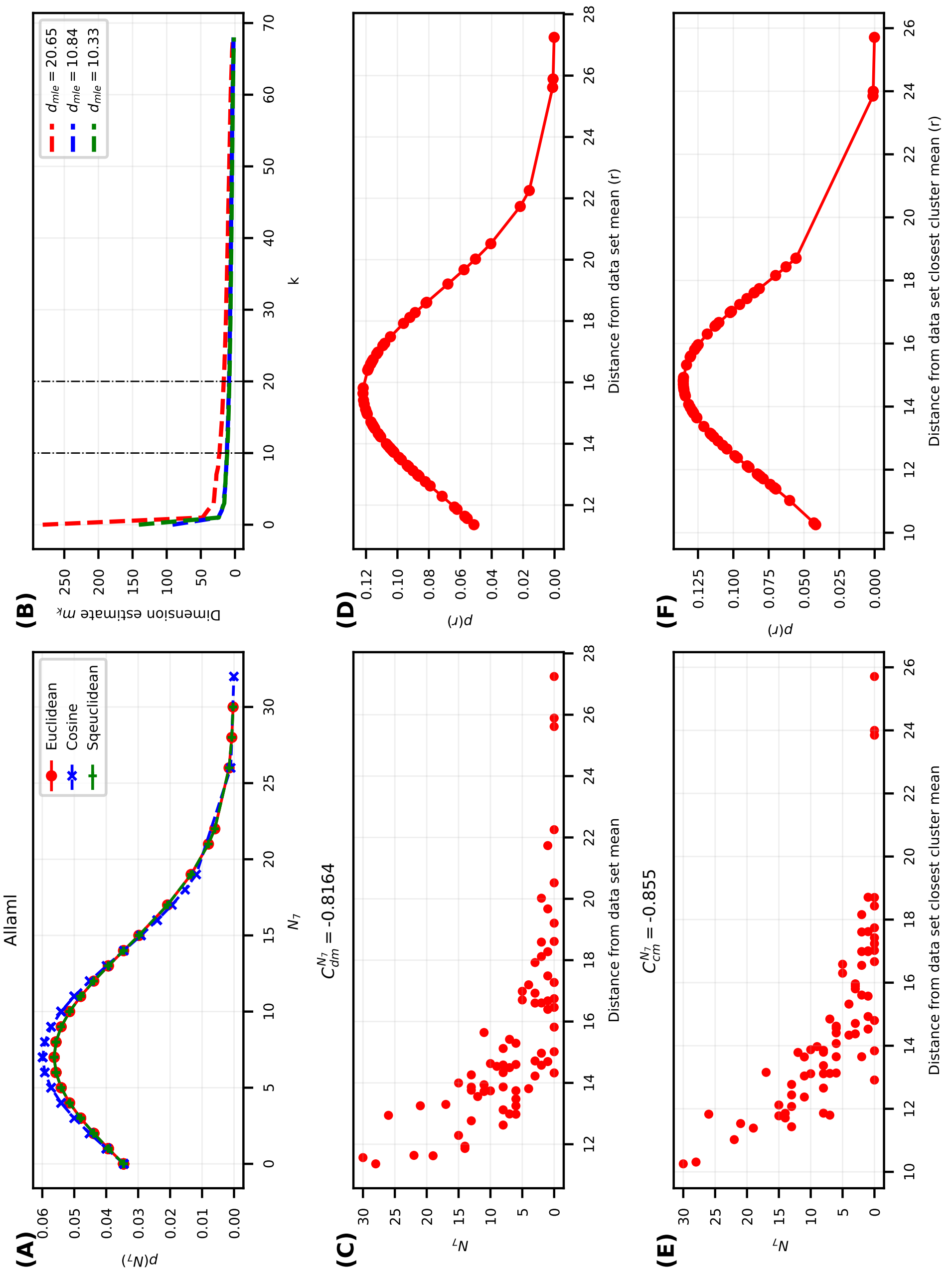


Figure C.1: Hubness analysis of Allaml dataset. Six indicators have been used to evaluate the hubness phenomenon: (A) empirical distribution of N_7 for Euclidean, square Euclidean and cosine distances; (B) intrinsic dimensionality analysis; (C) scatter plot and spearman correlation of N_7 against the Euclidean distance to the dataset mean; (D) probability density function of observing point at distance r from the mean of the dataset; (E) scatter plot and spearman correlation of N_7 against the Euclidean distance to the closets cluster mean; (F) probability density function of observing point at distance r from the closets cluster mean.

C.2 Arcene

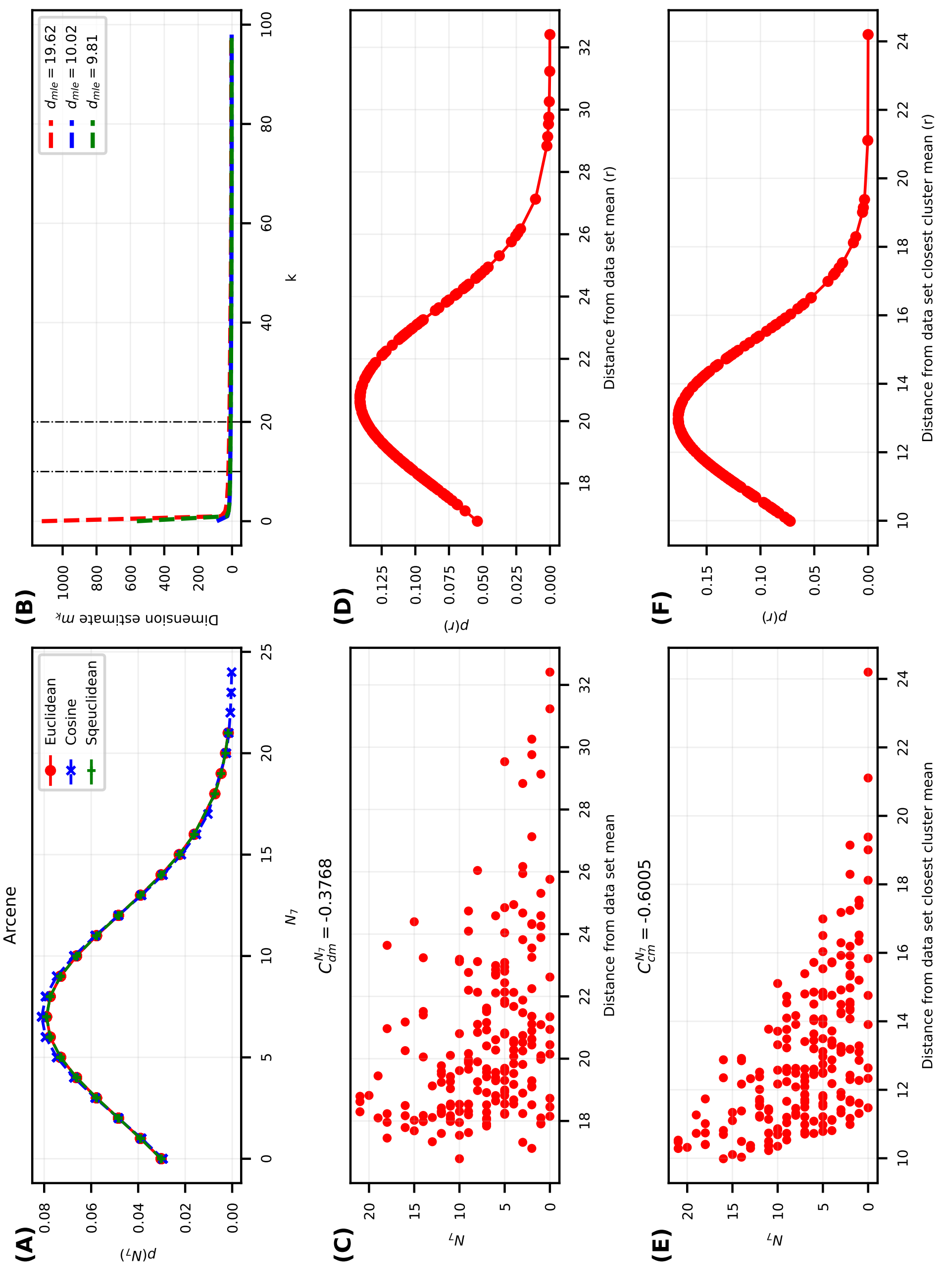


Figure C.2: Hubness analysis of Arcene dataset. Six indicators have been used to evaluate the hubness phenomenon: (A) empirical distribution of N_7 for Euclidean, square Euclidean and cosine distances; (B) intrinsic dimensionality analysis; (C) scatter plot and spearman correlation of N_7 against the Euclidean distance to the dataset mean; (D) probability density function of observing point at distance r from the mean of the dataset; (E) scatter plot and spearman correlation of N_7 against the Euclidean distance to the closets cluster mean; (F) probability density function of observing point at distance r from the closets cluster mean.

C.3 Basehock

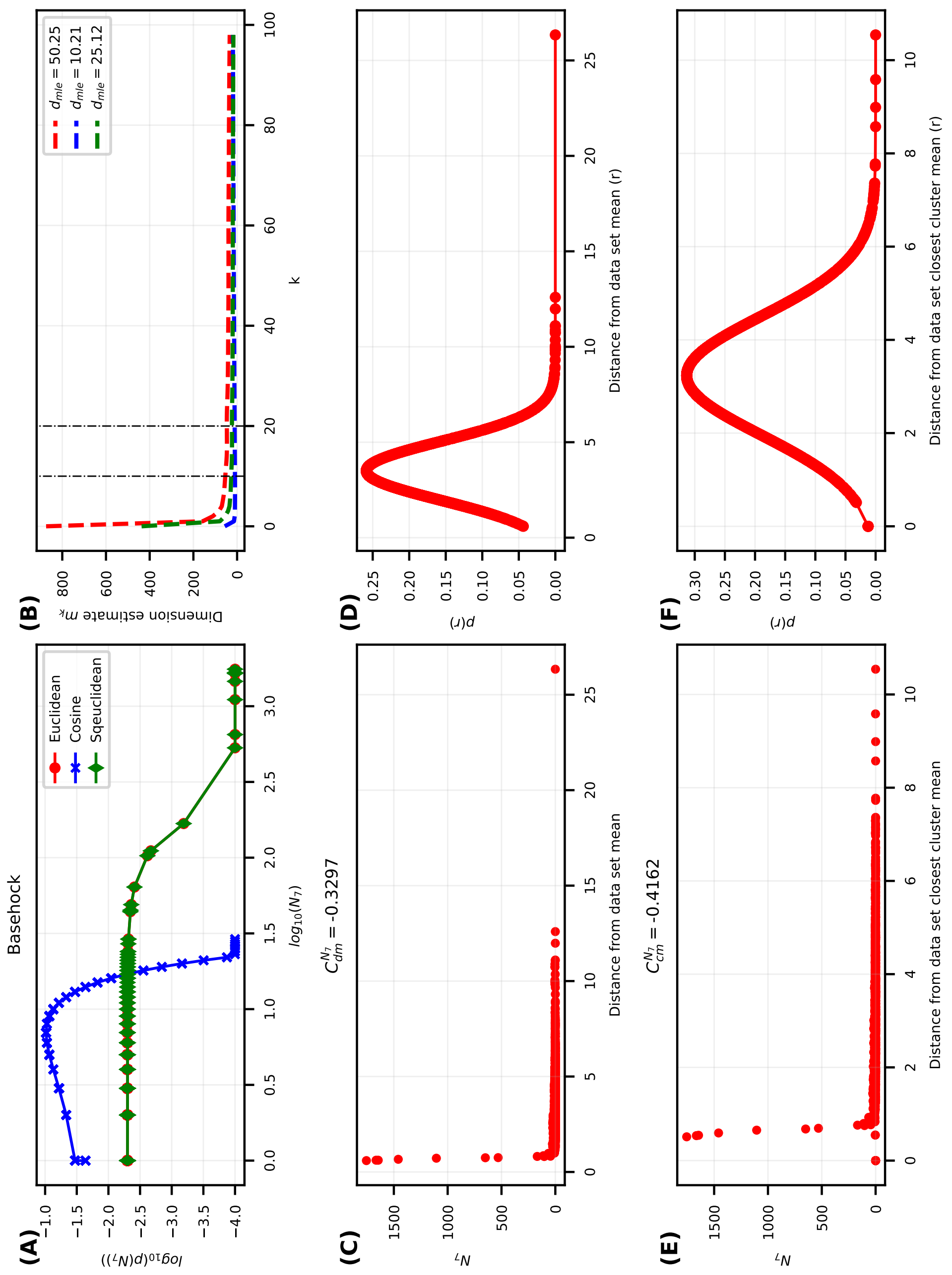


Figure C.3: Hubness analysis of Basehock dataset. Six indicators have been used to evaluate the hubness phenomenon: (A) empirical distribution of N_7 for Euclidean, square Euclidean and cosine distances; (B) intrinsic dimensionality analysis; (C) scatter plot and spearman correlation of N_7 against the Euclidean distance to the dataset mean; (D) probability density function of observing point at distance r from the mean of the dataset; (E) scatter plot and spearman correlation of N_7 against the Euclidean distance to the closets cluster mean; (F) probability density function of observing point at distance r from the closets cluster mean.

C.4 Colon

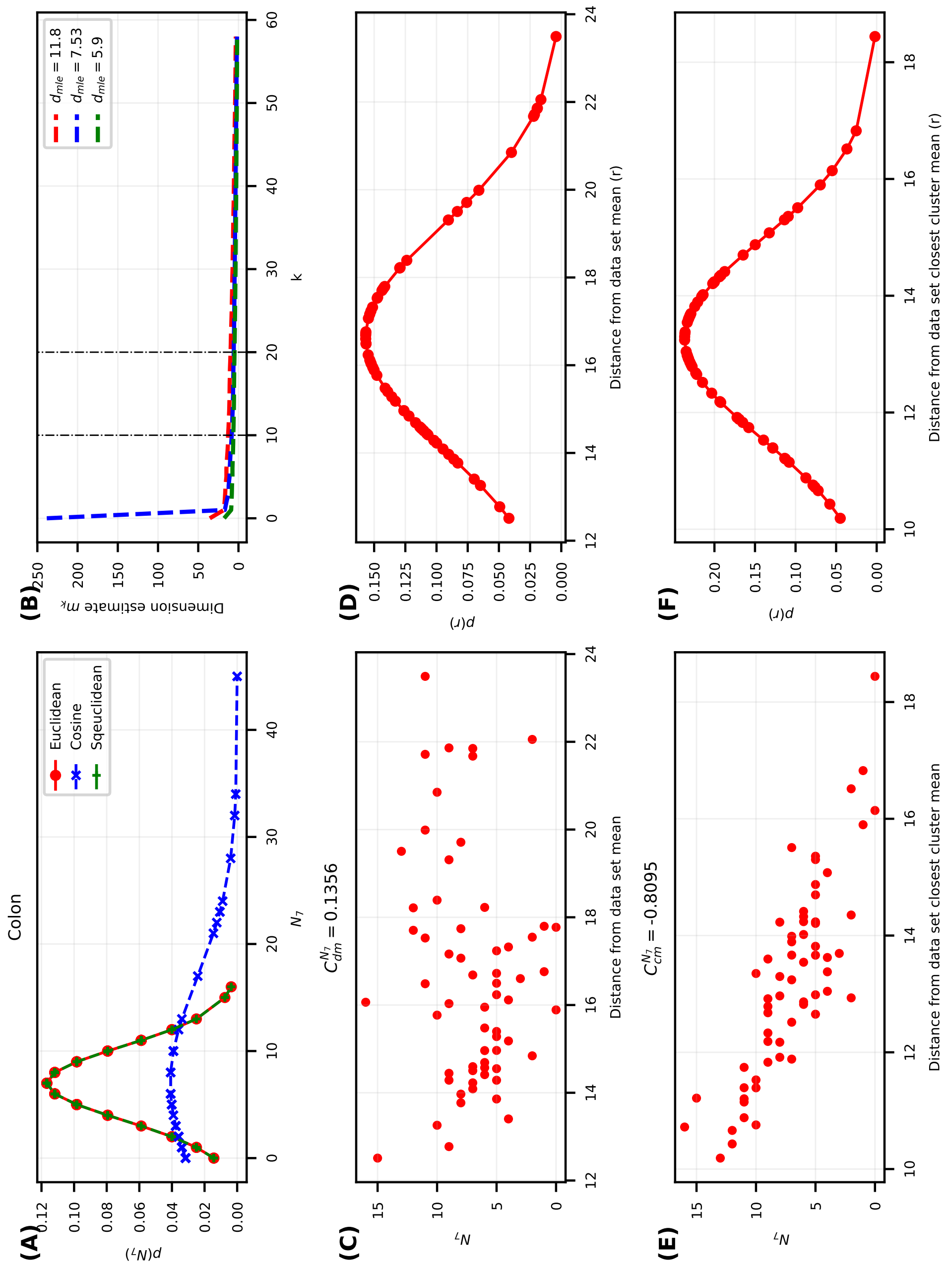


Figure C.4: Hubness analysis of Colon dataset. Six indicators have been used to evaluate the hubness phenomenon: (A) empirical distribution of N_7 for Euclidean, square Euclidean and cosine distances; (B) intrinsic dimensionality analysis; (C) scatter plot and spearman correlation of N_7 against the Euclidean distance to the dataset mean; (D) probability density function of observing point at distance r from the mean of the dataset; (E) scatter plot and spearman correlation of N_7 against the Euclidean distance to the closets cluster mean; (F) probability density function of observing point at distance r from the closets cluster mean.

C.5 Gli

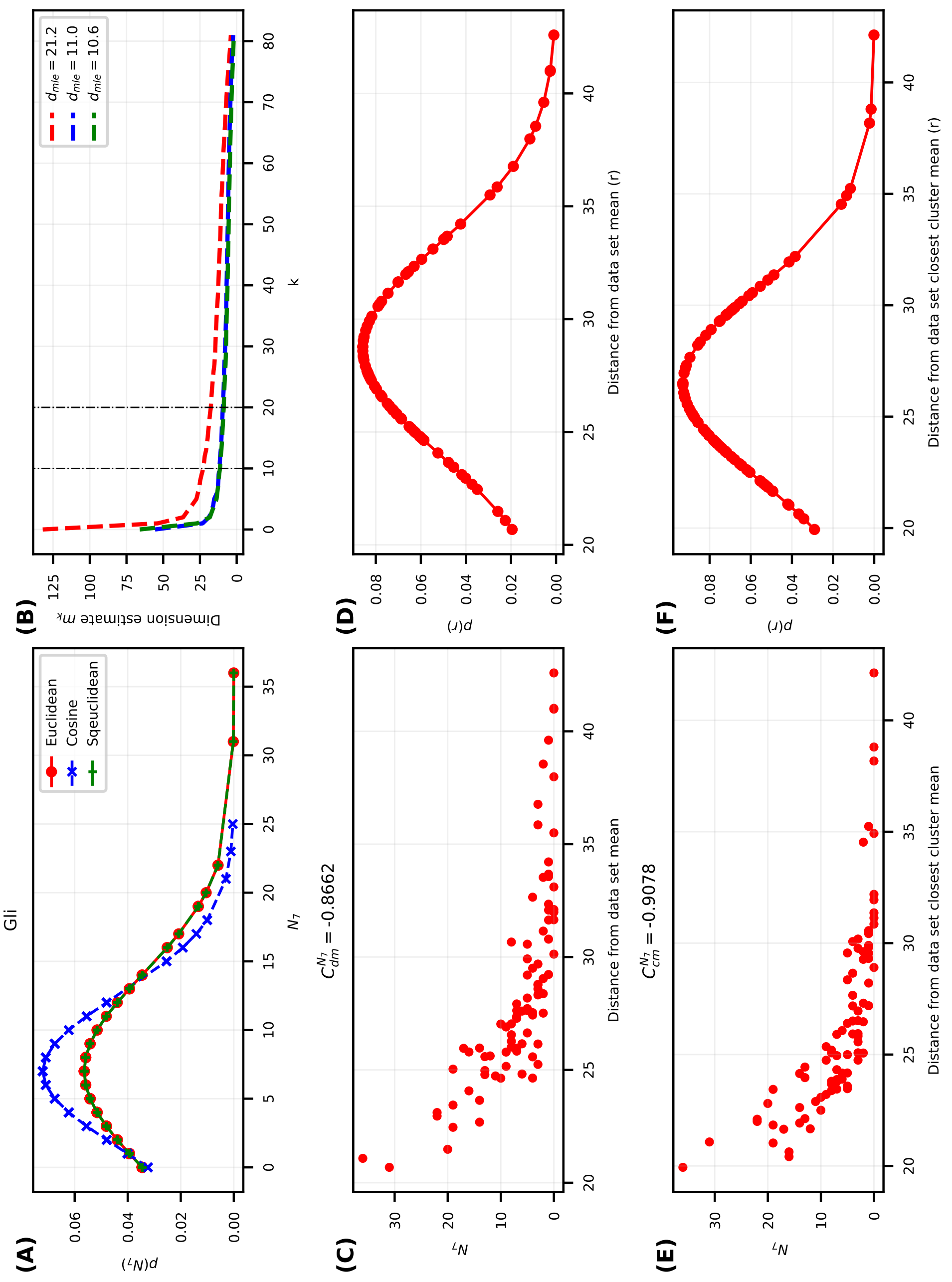


Figure C.5: Hubness analysis of Gli dataset. Six indicators have been used to evaluate the hubness phenomenon: (A) empirical distribution of N_7 for Euclidean, square Euclidean and cosine distances; (B) intrinsic dimensionality analysis; (C) scatter plot and spearman correlation of N_7 against the Euclidean distance to the dataset mean; (D) probability density function of observing point at distance r from the mean of the dataset; (E) scatter plot and spearman correlation of N_7 against the Euclidean distance to the closets cluster mean; (F) probability density function of observing point at distance r from the closets cluster mean.

C.6 Leukemia

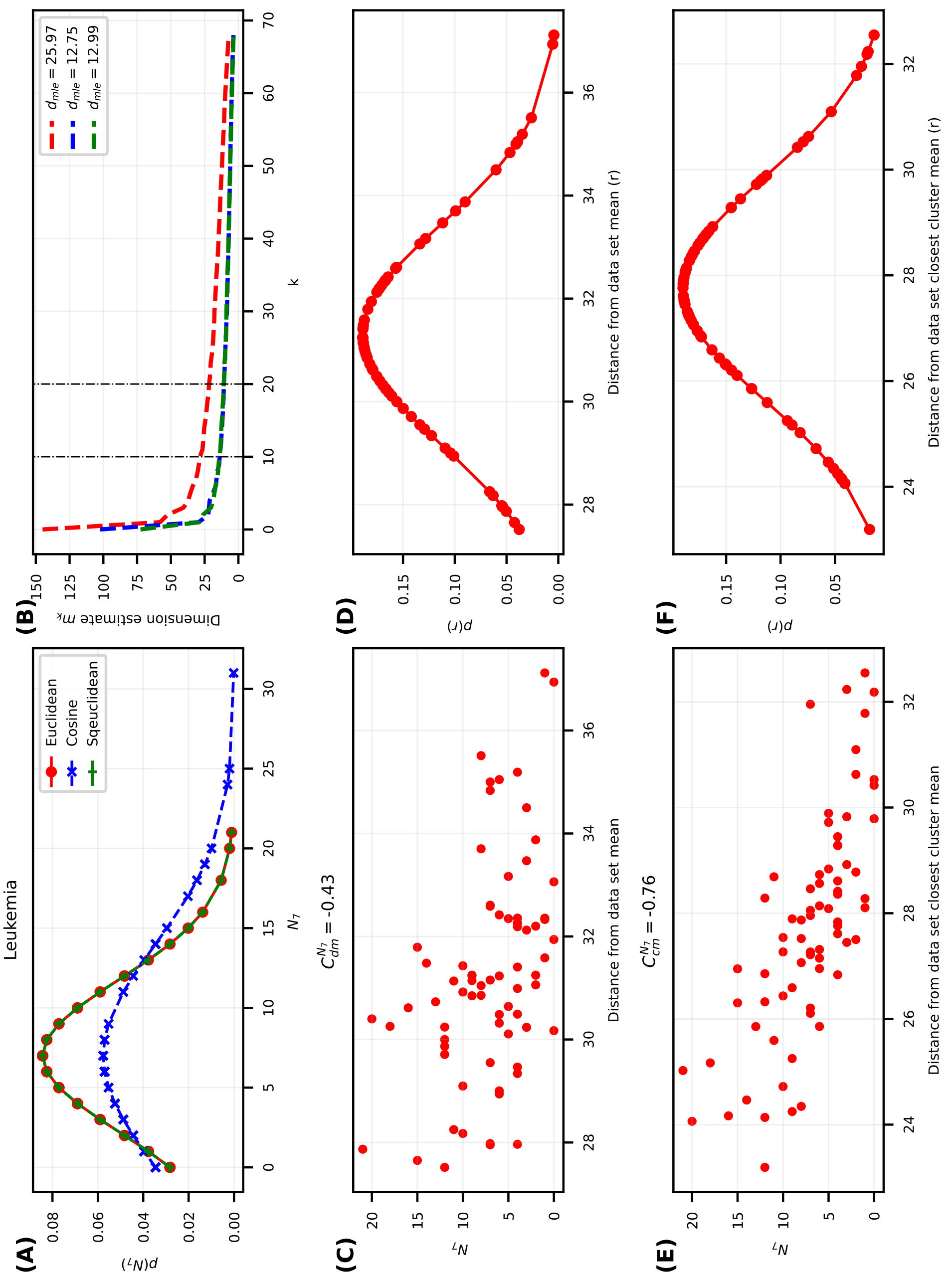


Figure C.6: Hubness analysis of Leukemia dataset. Six indicators have been used to evaluate the hubness phenomenon: (A) empirical distribution of N_7 for Euclidean, square Euclidean and cosine distances; (B) intrinsic dimensionality analysis; (C) scatter plot and spearman correlation of N_7 against the Euclidean distance to the dataset mean; (D) probability density function of observing point at distance r from the mean of the dataset; (E) scatter plot and spearman correlation of N_7 against the Euclidean distance to the closets cluster mean; (F) probability density function of observing point at distance r from the closets cluster mean.

C.7 Pcmac

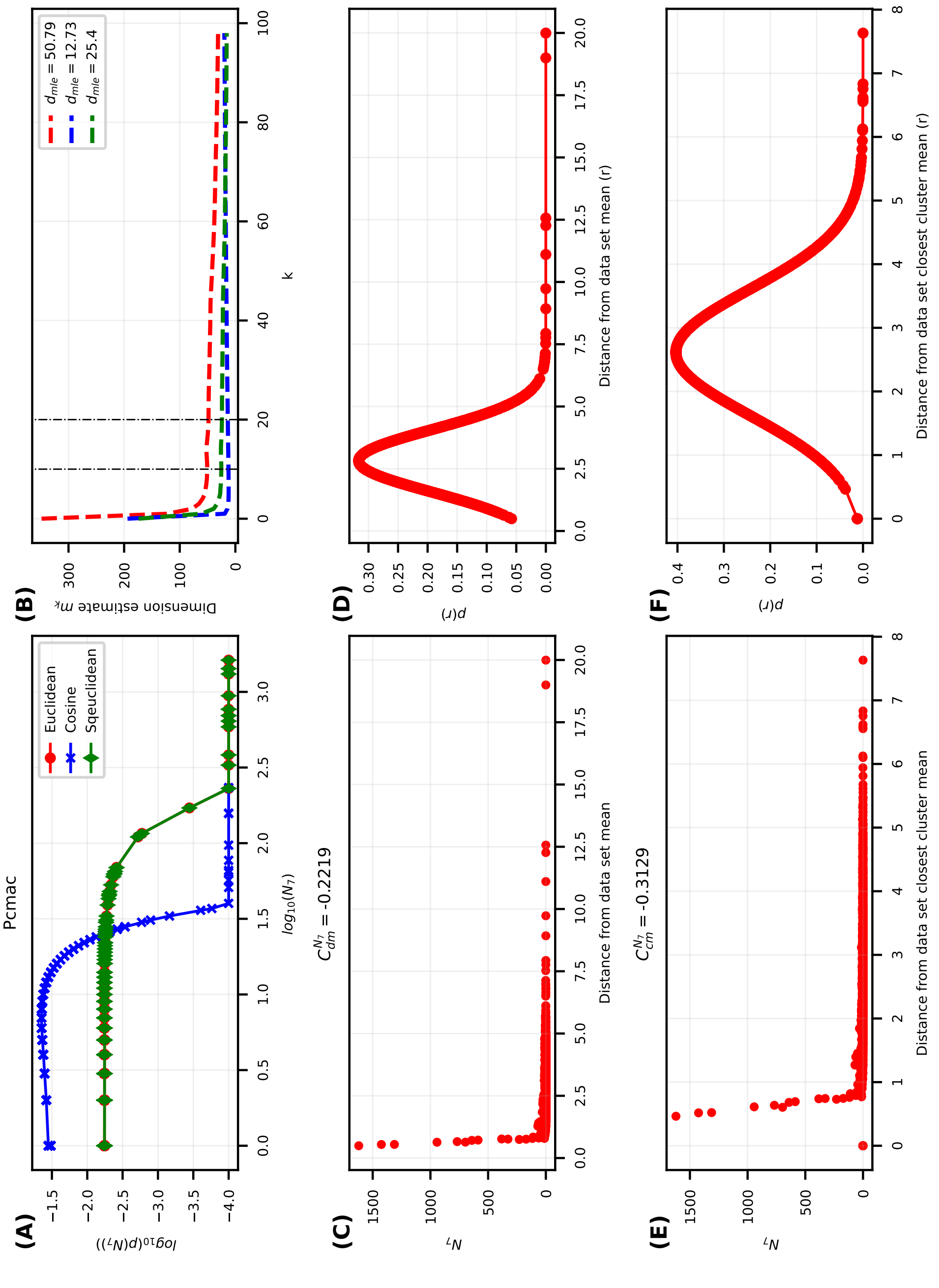


Figure C.7: Hubness analysis of Pcmac dataset. Six indicators have been used to evaluate the hubness phenomenon: (A) empirical distribution of N_7 for Euclidean, square Euclidean and cosine distances; (B) intrinsic dimensionality analysis; (C) scatter plot and spearman correlation of N_7 against the Euclidean distance to the dataset mean; (D) probability density function of observing point at distance r from the mean of the dataset; (E) scatter plot and spearman correlation of N_7 against the Euclidean distance to the closets cluster mean; (F) probability density function of observing point at distance r from the closets cluster mean.

C.8 Relathe

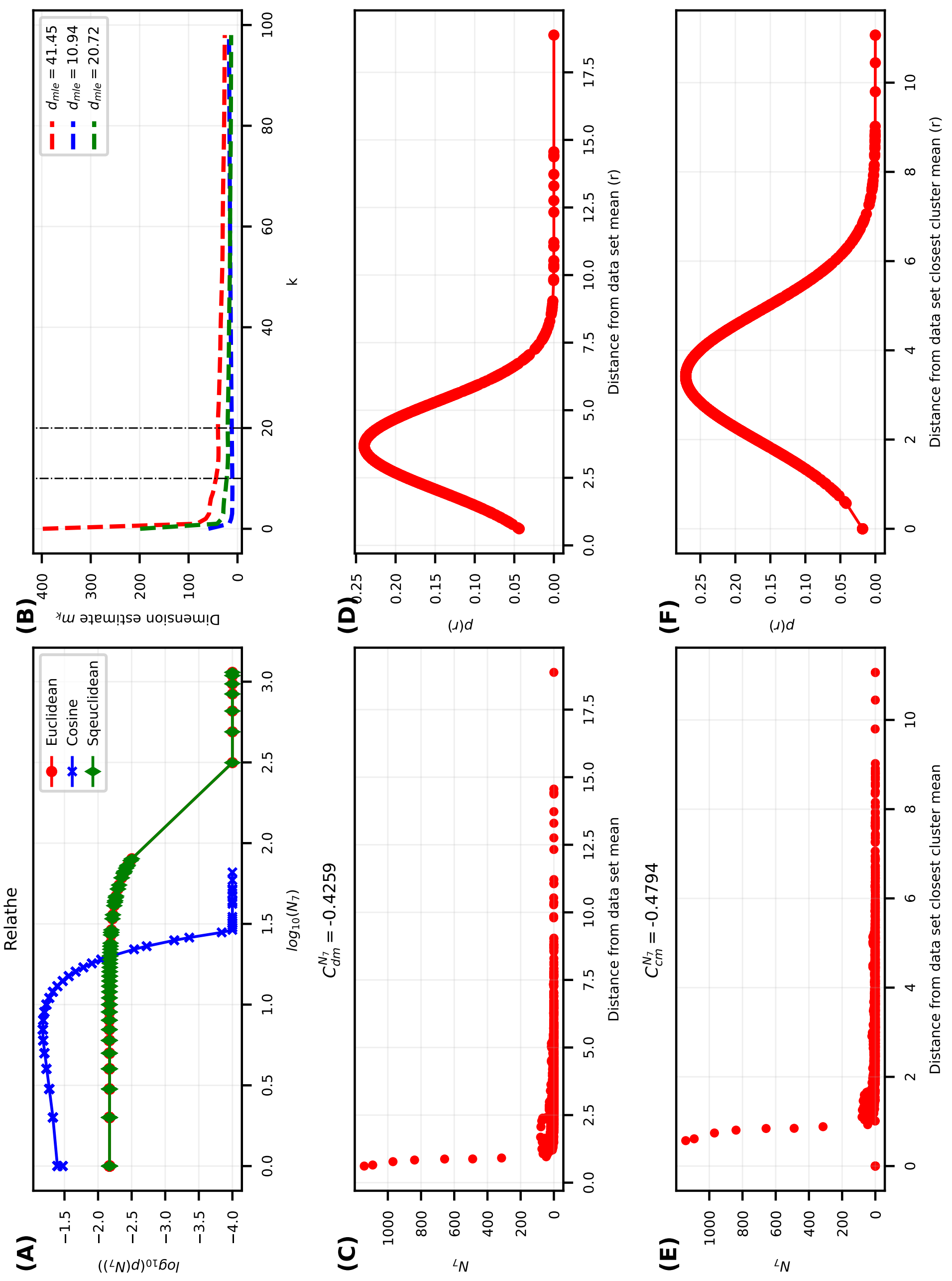


Figure C.8: Hubness analysis of Relathe dataset. Six indicators have been used to evaluate the hubness phenomenon: (A) empirical distribution of N_7 for Euclidean, square Euclidean and cosine distances; (B) intrinsic dimensionality analysis; (C) scatter plot and spearman correlation of N_7 against the Euclidean distance to the dataset mean; (D) probability density function of observing point at distance r from the mean of the dataset; (E) scatter plot and spearman correlation of N_7 against the Euclidean distance to the closets cluster mean; (F) probability density function of observing point at distance r from the closets cluster mean.

C.9 Smkcan

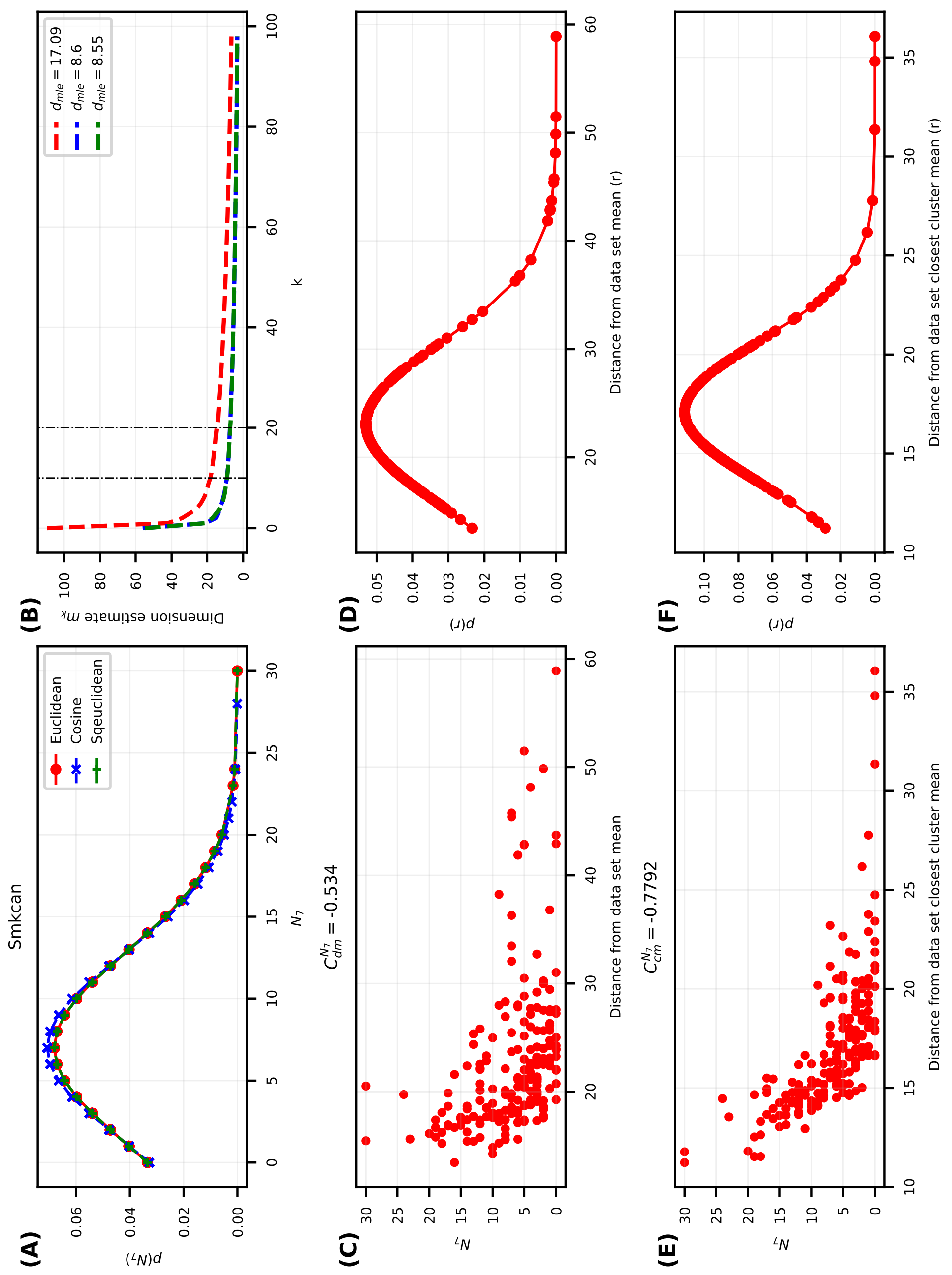


Figure C.9: Hubness analysis of Smkcan dataset. Six indicators have been used to evaluate the hubness phenomenon: (A) empirical distribution of N_7 for Euclidean, square Euclidean and cosine distances; (B) intrinsic dimensionality analysis; (C) scatter plot and spearman correlation of N_7 against the Euclidean distance to the dataset mean; (D) probability density function of observing point at distance r from the mean of the dataset; (E) scatter plot and spearman correlation of N_7 against the Euclidean distance to the closets cluster mean; (F) probability density function of observing point at distance r from the closets cluster mean.

Appendix D

Feature Importance Analysis

***

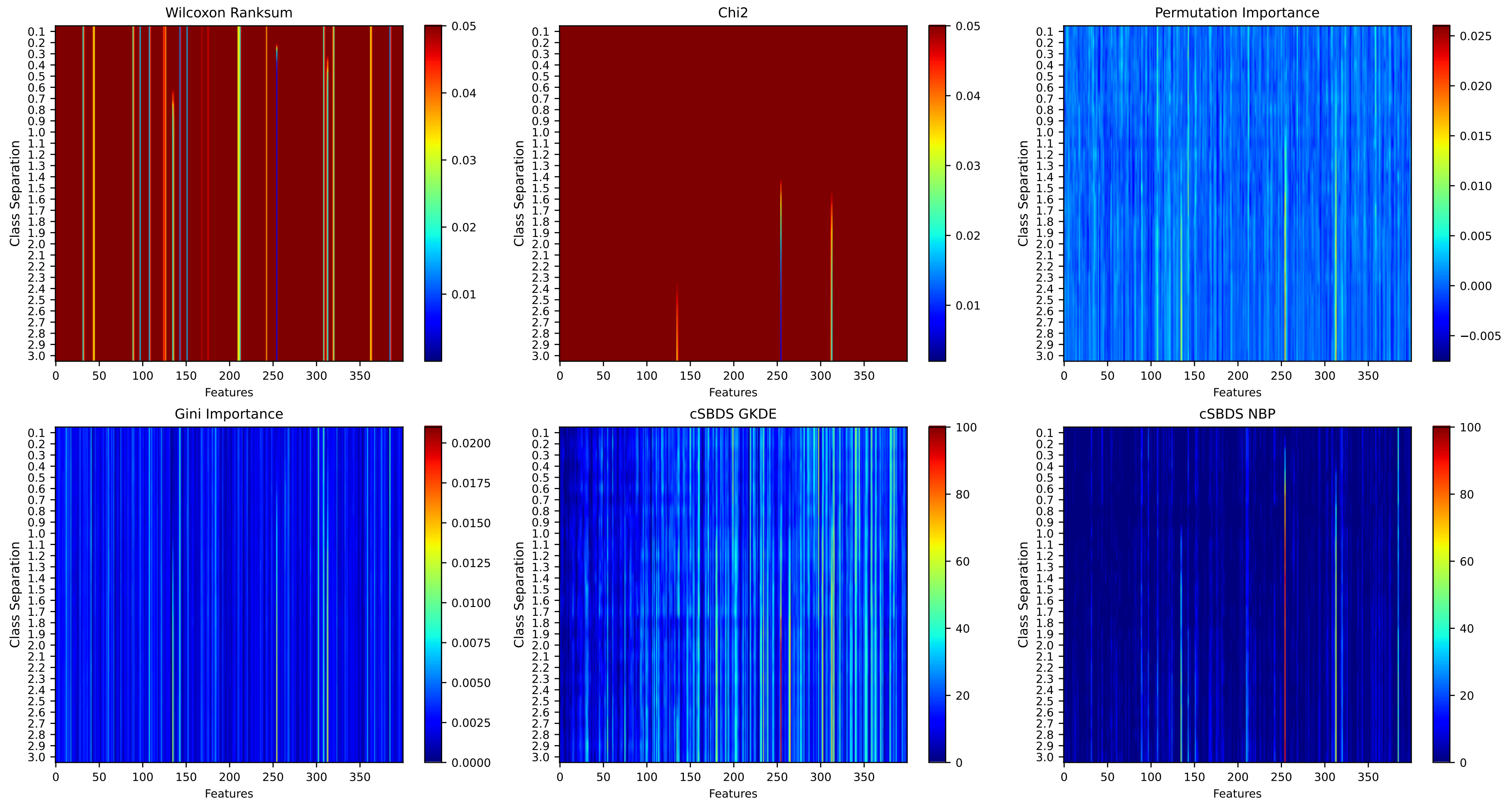


Figure D.1: Feature importance methods comparison using synthetically created datasets varying the class separation parameter from 0.1 to 3 and with 2 cluster per class. For the Wilcoxon ranksum test and the chi-square test the p-value is shown (the p-value higher than 0.05 was set to 0.05 to simplify the visualisation). The cSBDS GKDE and the cSBDS NBP show the frequency of appearance of the features in the subspaces selected for the classification.

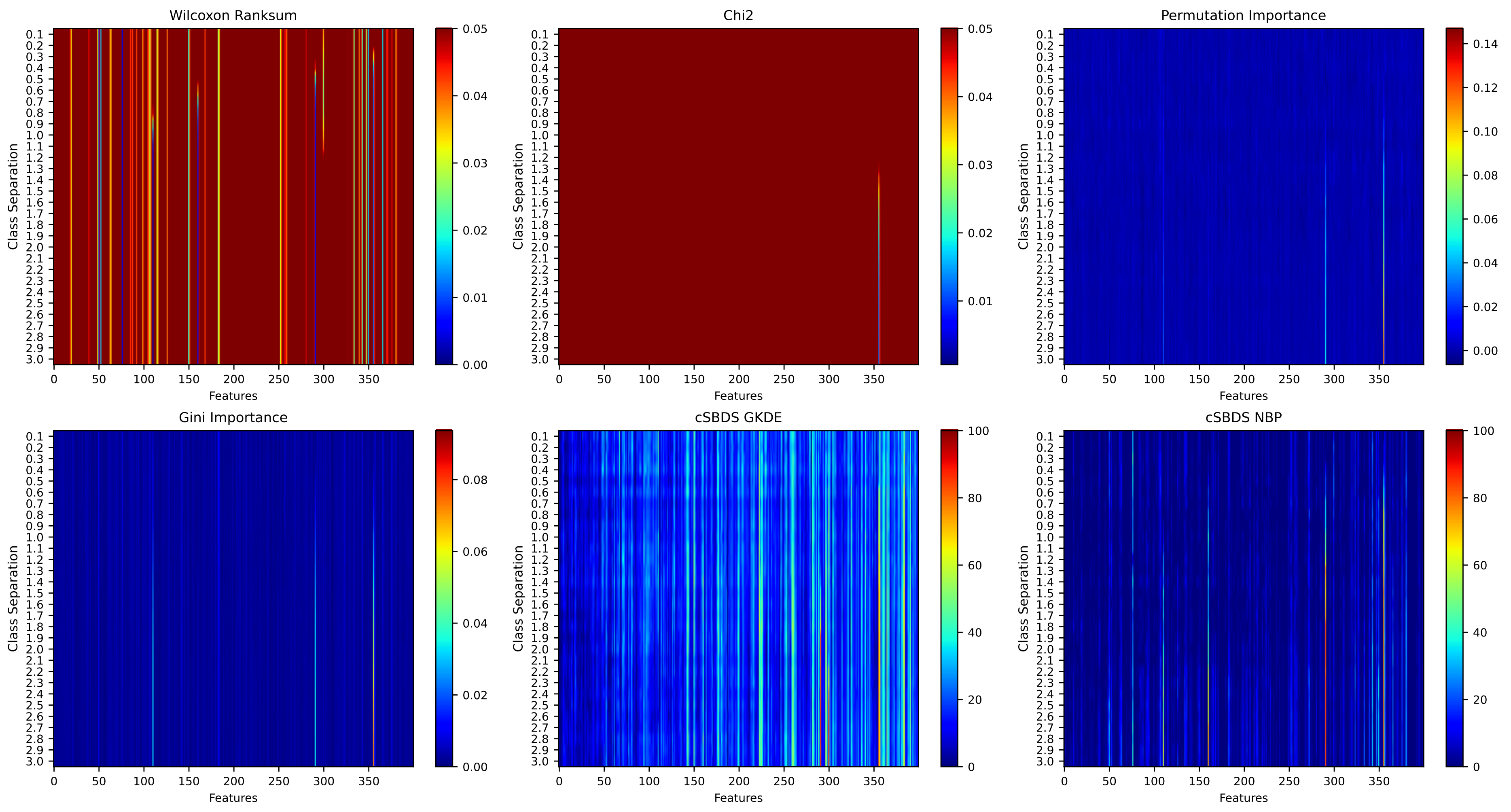


Figure D.2: Feature importance methods comparison using synthetically created datasets varying the class separation parameter from 0.1 to 3 and with 3 cluster per class. For the Wilcoxon ranksum test and the chi-square test the p-value is shown (the p-value higher than 0.05 was set to 0.05 to simplify the visualisation). The cSBDS GKDE and the cSBDS NBP show the frequency of appearance of the features in the subspaces selected for the classification.

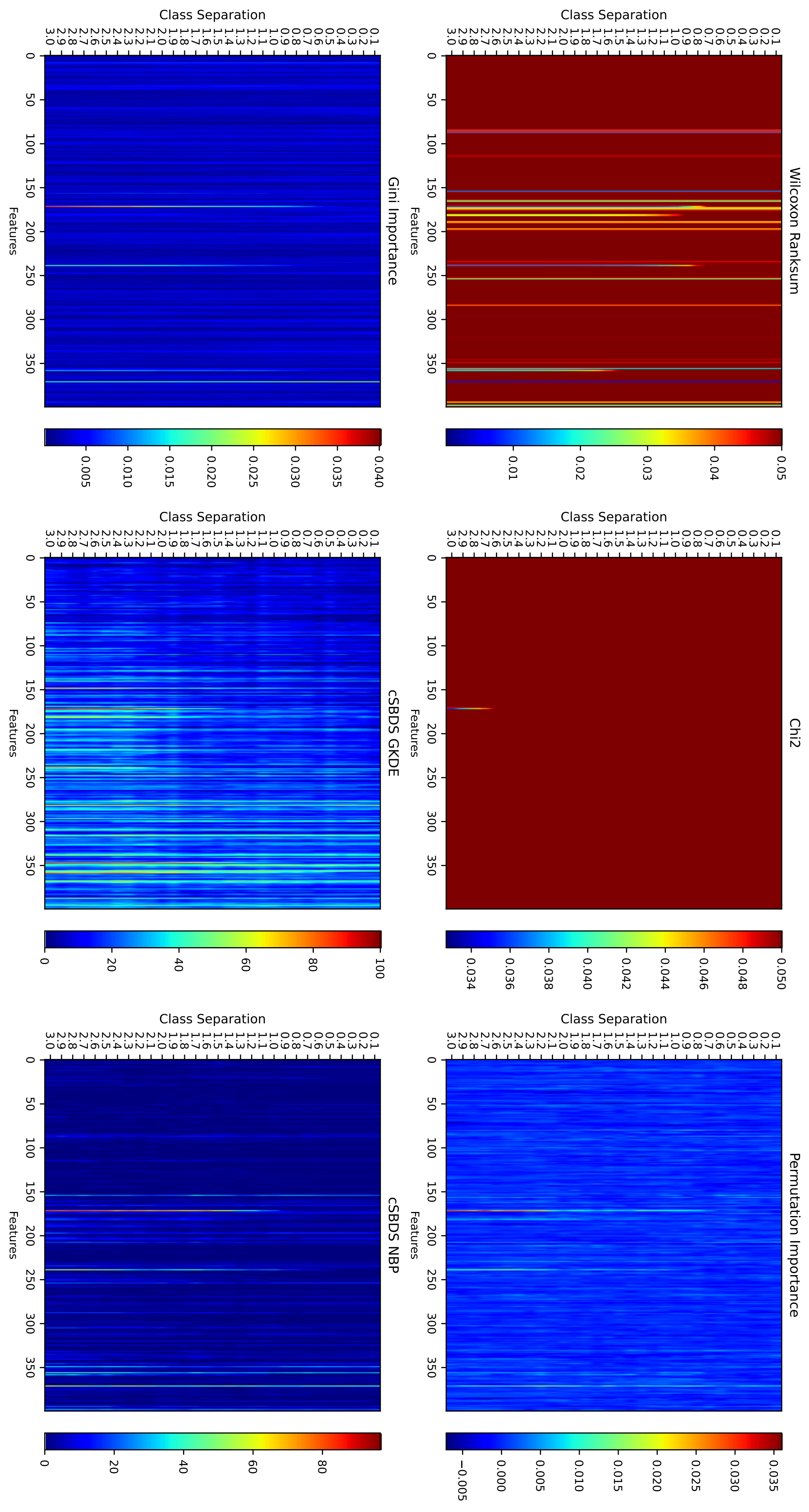


Figure D.3: Feature importance methods comparison using synthetically created datasets varying the class separation parameter from 0.1 to 3 and with 4 cluster per class. For the Wilcoxon ranksum test and the chi-square test the p-value is shown (the p-value higher than 0.05 was set to 0.05 to simplify the visualisation). The cSBDS GKDE and the cSBDS NBP show the frequency of appearance of the features in the subspaces selected for the classification.

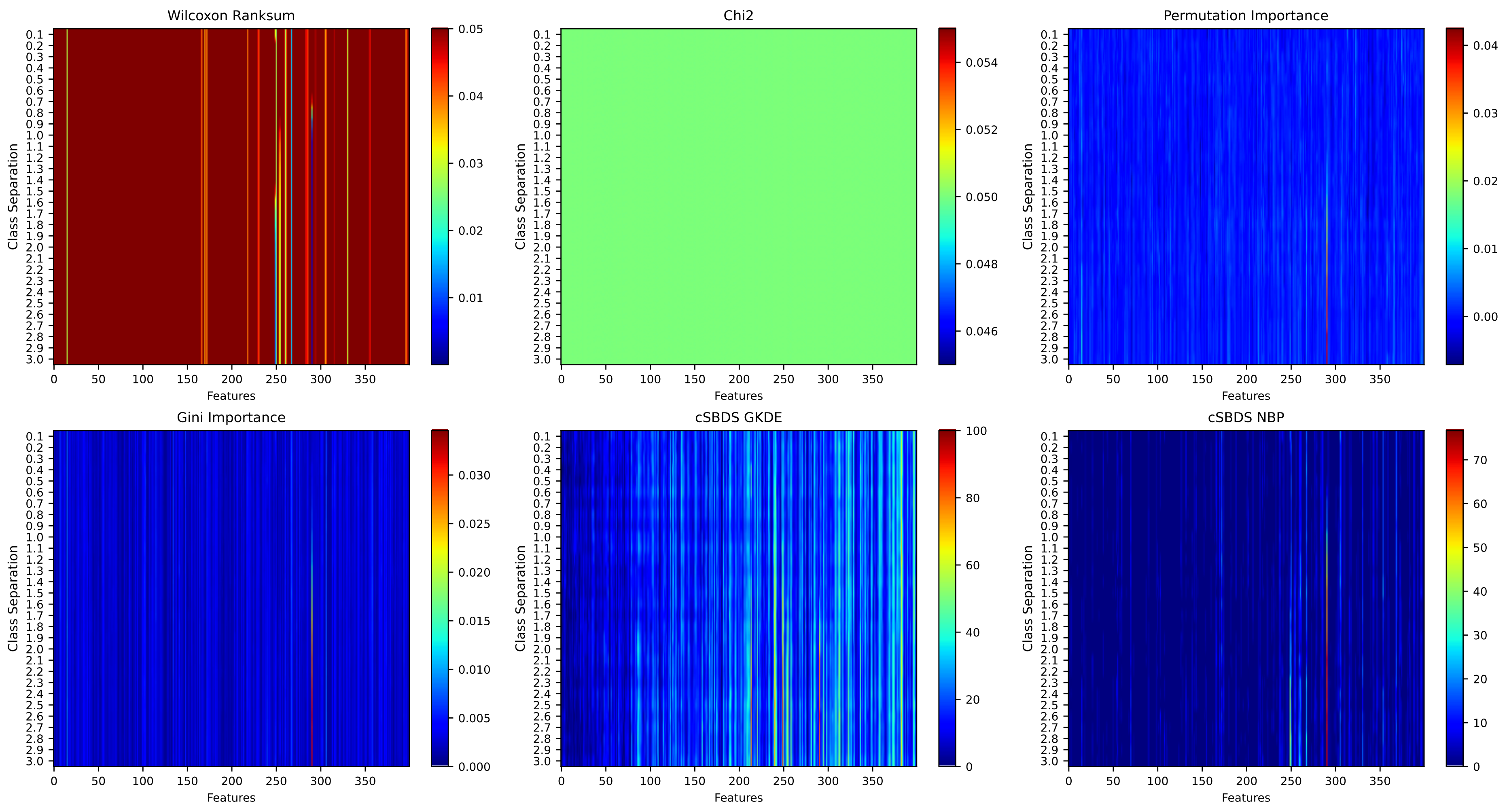


Figure D.4: Feature importance methods comparison using synthetically created datasets varying the class separation parameter from 0.1 to 3 and with 5 cluster per class. For the Wilcoxon ranksum test and the chi-square test the p-value is shown (the p-value higher than 0.05 was set to 0.05 to simplify the visualisation). The cSBDS GKDE and the cSBDS NBP show the frequency of appearance of the features in the subspaces selected for the classification.

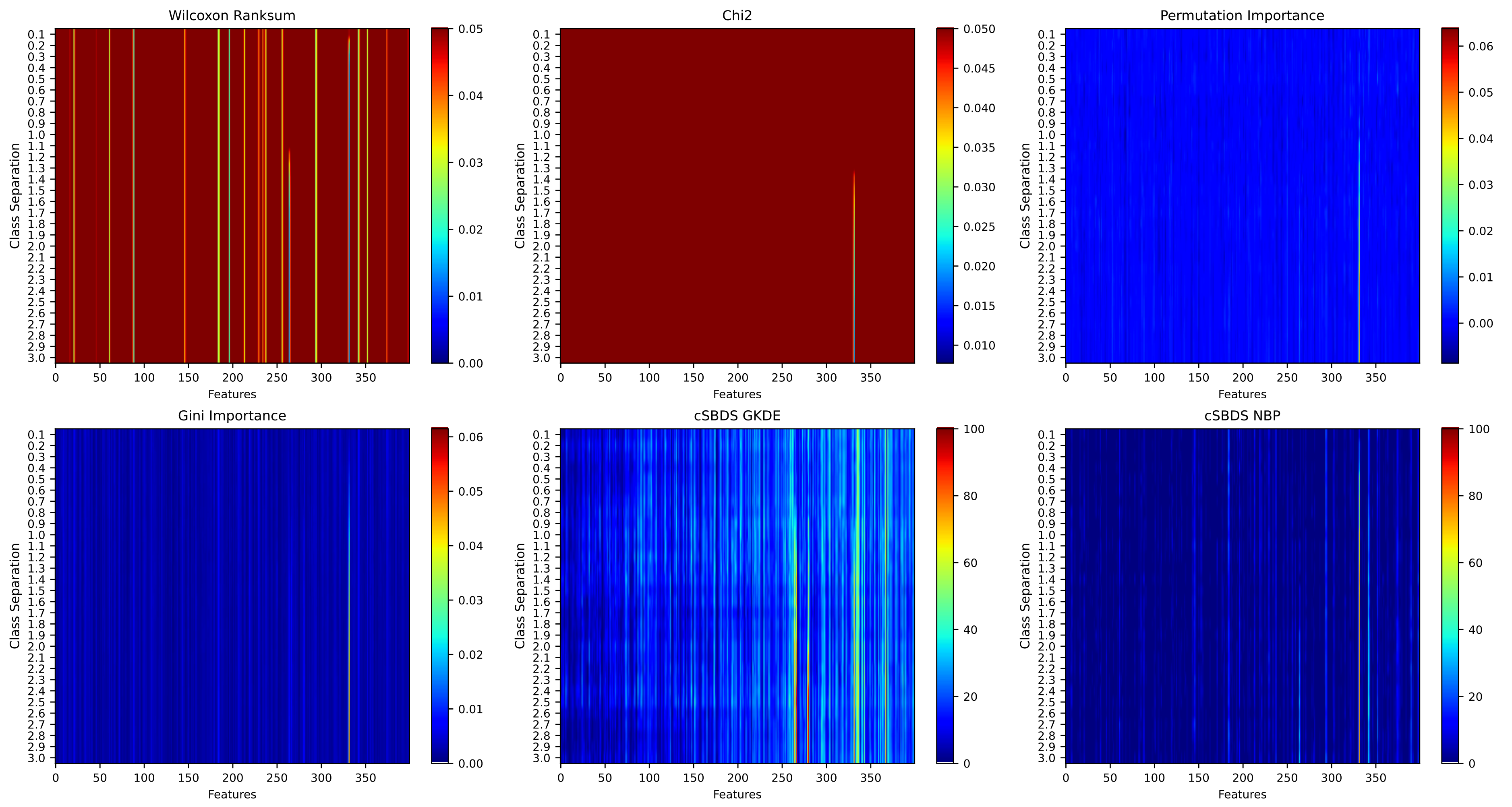


Figure D.5: Feature importance methods comparison using synthetically created datasets varying the class separation parameter from 0.1 to 3 and with 6 cluster per class. For the Wilcoxon ranksum test and the chi-square test the p-value is shown (the p-value higher than 0.05 was set to 0.05 to simplify the visualisation). The cSBDS GKDE and the cSBDS NBP show the frequency of appearance of the features in the subspaces selected for the classification.

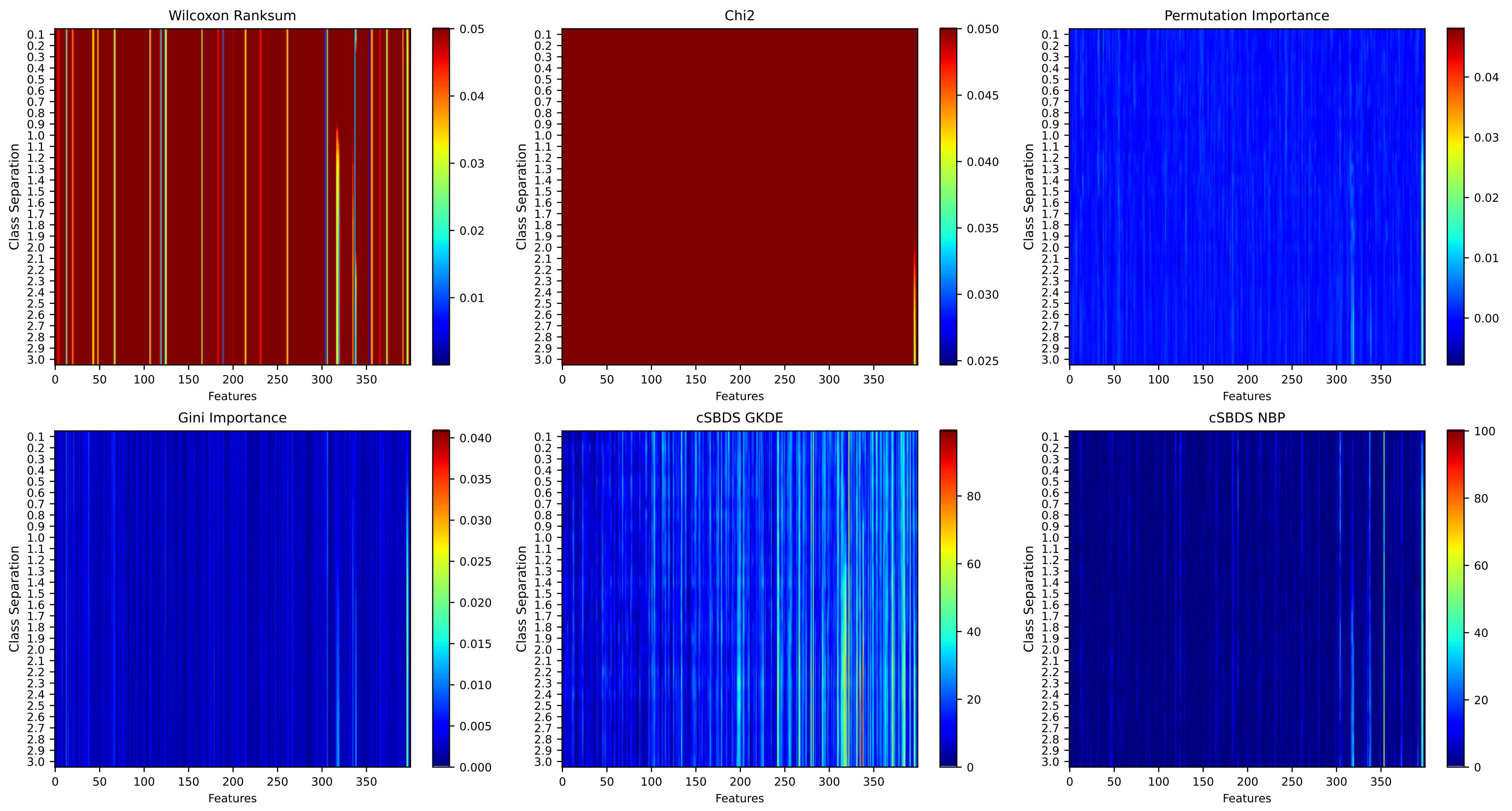


Figure D.6: Feature importance methods comparison using synthetically created datasets varying the class separation parameter from 0.1 to 3 and with 7 cluster per class. For the Wilcoxon ranksum test and the chi-square test the p-value is shown (the p-value higher than 0.05 was set to 0.05 to simplify the visualisation). The cSBDS GKDE and the cSBDS NBP show the frequency of appearance of the features in the subspaces selected for the classification.

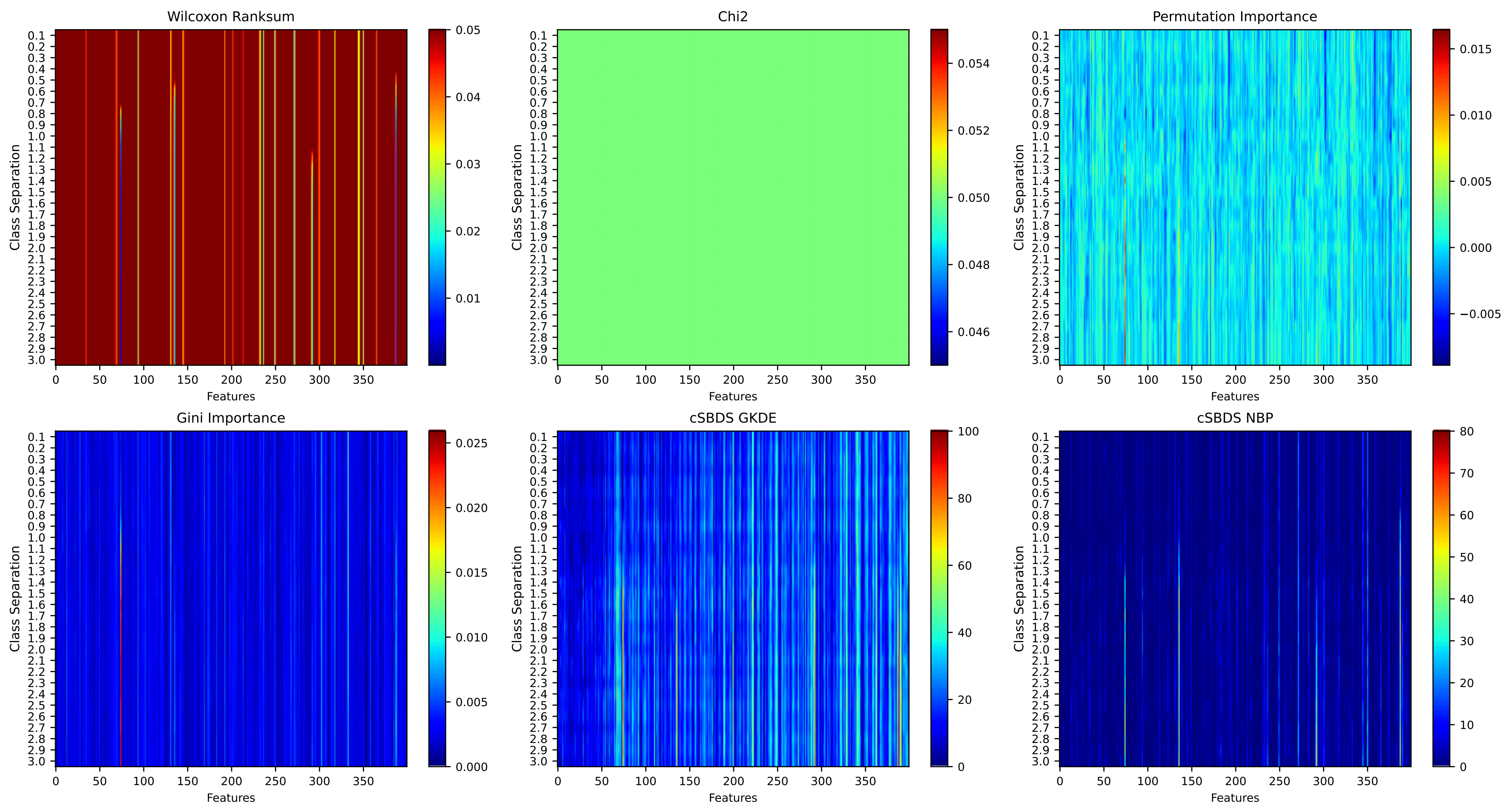


Figure D.7: Feature importance methods comparison using synthetically created datasets varying the class separation parameter from 0.1 to 3 and with 8 cluster per class. For the Wilcoxon ranksum test and the chi-square test the p-value is shown (the p-value higher than 0.05 was set to 0.05 to simplify the visualisation). The cSBDS GKDE and the cSBDS NBP show the frequency of appearance of the features in the subspaces selected for the classification.

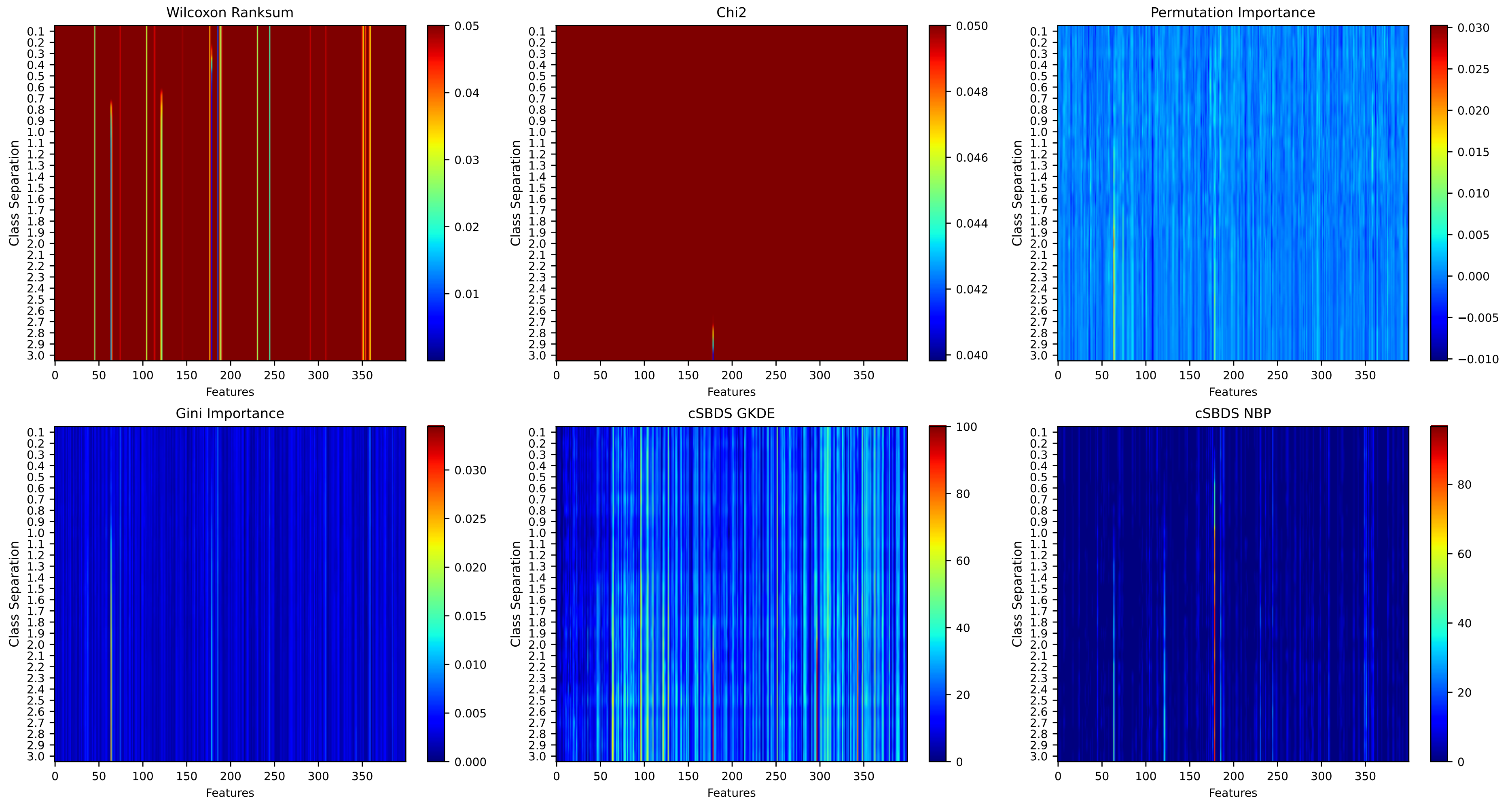


Figure D.8: Feature importance methods comparison using synthetically created datasets varying the class separation parameter from 0.1 to 3 and with 9 cluster per class. For the Wilcoxon ranksum test and the chi-square test the p-value is shown (the p-value higher than 0.05 was set to 0.05 to simplify the visualisation). The cSBDS GKDE and the cSBDS NBP show the frequency of appearance of the features in the subspaces selected for the classification.

Bibliography

***

[1] M. J. Zaki, W. Meira Jr, and W. Meira, *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, 2014.

[2] C. C. Aggarwal *et al.*, *Data mining: the textbook*. Springer, 2015, vol. 1.

[3] L. I. Kuncheva, "Switching between selection and fusion in combining classifiers: An experiment," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 32, no. 2, pp. 146–156, Apr. 2002, ISSN: 1083-4419. DOI: 10.1109/3477.990871.

[4] R. M. Cruz, R. Sabourin, and G. D. Cavalcanti, "Dynamic classifier selection: Recent advances and perspectives," *Information Fusion*, vol. 41, no. Supplement C, pp. 195–216, 2018, ISSN: 1566-2535. DOI: 10.1016/j.inffus.2017.09.010.

[5] R. M. Cruz, H. H. Zakane, R. Sabourin, and G. D. Cavalcanti, "Dynamic ensemble selection vs k-nn: Why and when dynamic selection obtains higher classification performance?" In *The Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, (Nov. 28–Dec. 1, 2017), Montreal, Canada, 2017.

[6] A. S. Britto Jr., R. Sabourin, and L. E. S. Oliveira, "Dynamic selection of classifiers - a comprehensive review," *Pattern Recogn.*, vol. 47, no. 11, pp. 3665–3680, Nov. 2014, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2014.05.003.

[7] A. H. R. Ko, R. Sabourin, and A. S. Britto Jr., "From dynamic classifier selection to dynamic ensemble selection," *Pattern Recogn.*, vol. 41, no. 5, pp. 1718–1731, May 2008, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2007.10.015.

[8] A. Maciel-Guerra, G. P. Figueredo, F. J. V. Zuben, E. Marti, J. Twycross, and M. J. C. Alcocer, “Microarray feature selection and dynamic selection of classifiers for early detection of insect bite hypersensitivity in horses,” in *IEEE Congress on Evolutionary Computation, CEC 2019, Wellington, New Zealand, June 10-13, 2019*, IEEE, 2019, pp. 1157–1164. DOI: 10.1109/CEC.2019.8790319.

[9] A. Maciel-Guerra, G. P. Figueredo, and J. Twycross, “Dynamic selection of classifiers applied to high-dimensional small-instance data sets: Problems and challenges,” in *LOD2020 - The Sixth International Conference on Machine Learning, Optimization, and Data Science – July 19-23, 2020 – Certosa di Pontignano, Siena – Tuscany, Italy*, Lecture Notes in Computer Science - LNCS, 2020. DOI: 10.1007/978-3-030-64583-0_56.

[10] L. Parsons, E. Haque, and H. Liu, “Subspace clustering for high dimensional data: A review,” *SIGKDD Explorations*, vol. 6, pp. 90–105, 2004. DOI: 10.1145/1007730.1007731.

[11] D. Agrawal, P. Bernstein, E. Bertino, S. Davidson, U. Dayal, M. Franklin, ..., and J. Widom, “Challenges and opportunities with big data: A white paper prepared for the computing community consortium committee of the computing research association,” *Computing Research Association*, 2012.

[12] C. Ballard and W. Wang, “Dynamic ensemble selection methods for heterogeneous data mining,” in *2016 12th World Congress on Intelligent Control and Automation (WCICA)*, Jun. 2016, pp. 1021–1026. DOI: 10.1109/WCICA.2016.7578244.

[13] D. L. Donoho, “High-dimensional data analysis: The curses and blessings of dimensionality,” in *AMS Conference on Math Challenges of the 21st century*, 2000, pp. 1–33.

[14] T. Pham, T. Tran, D. Phung, and S. Venkatesh, “Predicting healthcare trajectories from medical records: A deep learning approach,” *Journal of Biomedical Informatics*, vol. 69, no. Supplement C, pp. 218–229, 2017, ISSN: 1532-0464. DOI: 10.1016/j.jbi.2017.04.001.

[15] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: Review, opportunities and challenges," *Briefings in Bioinformatics*, bbx044, 2017. DOI: 10.1093/bib/bbx044.

[16] V. Bolón-Canedo, N. Sánchez-Maroño, A. Alonso-Betanzos, J. M. Benítez, and F. Herrera, "A review of microarray datasets and applied feature selection methods," *Information Science*, vol. 282, pp. 111–135, Oct. 2014, ISSN: 0020-0255. DOI: 10.1016/j.ins.2014.05.042.

[17] M. Verleysen and D. François, "The curse of dimensionality in data mining and time series prediction," in *Proceedings of the 8th International Conference on Artificial Neural Networks: Computational Intelligence and Bioinspired Systems*, ser. IWANN'05, Barcelona, Spain: Springer-Verlag, 2005, pp. 758–770, ISBN: 978-3-540-26208-4. DOI: 10.1007/11494669_93.

[18] H. E. Kiziloz, "Classifier ensemble methods in feature selection," *Neurocomputing*, vol. 419, pp. 97–107, 2021.

[19] B. A. Powell, "How i learned to stop worrying and love the curse of dimensionality: An appraisal of cluster validation in high-dimensional spaces," *arXiv preprint arXiv:2201.05214*, 2022.

[20] M. Radovanovic, A. Nanopoulos, and M. Ivanovic, "Hubs in space: Popular nearest neighbors in high-dimensional data," *J. Mach. Learn. Res.*, vol. 11, pp. 2487–2531, Dec. 2010, ISSN: 1532-4435.

[21] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?" In *International conference on database theory*, Springer, 1999, pp. 217–235.

[22] R. J. Durrant and A. Kabán, "When is 'nearest neighbour'meaningful: A converse theorem and implications," *Journal of Complexity*, vol. 25, no. 4, pp. 385–397, 2009.

[23] P. Demartines, "Analyse de données par réseaux de neurones auto-organisés," PhD thesis, Grenoble INPG, 1994.

[24] A. Hinneburg, C. C. Aggarwal, and D. A. Keim, "What is the nearest neighbor in high dimensional spaces?" In *26th Internat. Conference on Very Large Databases*, 2000, pp. 506–515.

[25] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *Database Theory — ICDT 2001*, J. Van den Bussche and V. Vianu, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 420–434.

[26] D. François, V. Wertz, and M. Verleysen, "The concentration of fractional distances," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 7, pp. 873–886, 2007.

[27] A. Nanopoulos, M. Radovanović, and M. Ivanović, "How does high dimensionality affect collaborative filtering?" In *Proceedings of the third ACM conference on Recommender systems*, 2009, pp. 293–296.

[28] S. B.Meshram and S. M. Shinde, "A survey on ensemble methods for high dimensional data classification in biomedicine field," *International Journal of Computer Applications*, vol. 111, pp. 5–7, Feb. 2015. DOI: 10.5120/19580-1162.

[29] T. Woloszynski and M. Kurzynski, "A probabilistic model of classifier competence for dynamic ensemble selection," *Pattern Recogn.*, vol. 44, no. 10-11, pp. 2656–2668, Oct. 2011, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2011.03.020.

[30] R. M. Cruz, R. Sabourin, G. D. Cavalcanti, and T. I. Ren, "Meta-des: A dynamic ensemble selection framework using meta-learning," *Pattern Recognition*, vol. 48, no. 5, pp. 1925–1935, 2015, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2014.12.003.

[31] G. Brown, J. Wyatt, R. Harris, and X. Yao, "Diversity creation methods: A survey and categorisation," *Information fusion*, vol. 6, no. 1, pp. 5–20, 2005.

[32] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine learning*, vol. 51, no. 2, pp. 181–207, 2003.

[33] Y. Piao, M. Piao, C. H. Jin, H. S. Shon, J.-M. Chung, B. Hwang, and K. H. Ryu, "A New Ensemble Method with Feature Space Partitioning for High-Dimensional Data Classification," *Mathematical Problems in Engineering*, vol. 2015, no. i, pp. 1–12, 2015, ISSN: 1024-123X. DOI: 10.1155/2015/590678. [Online]. Available: http://www.hindawi.com/journals/mpe/2015/590678/.

[34] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, Aug. 1996, ISSN: 0885-6125. DOI: 10.1023/A:1018054314350.

[35] H. A. de Menezes Sabino Almeida, "Seleção dinâmica de classificadores baseada em filtragem e em distância adaptativa," Master's thesis, Federal University of Pernambuco, Recife, Brazil, 2014.

[36] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "Feature selection for high-dimensional data," *Progress in Artificial Intelligence*, vol. 5, no. 2, pp. 65–75, May 2016, ISSN: 2192-6360. DOI: 10.1007/s13748-015-0080-y.

[37] A. Tsymbal, S. Puuronen, and I. Skrypnyk, "Ensemble feature selection with dynamic integration of classifiers," in *International Congress on Comp. Intelligence Methods and Applications CIMA2001*, 2001. DOI: 10.1007/3-540-39963-1_44.

[38] M. Pechenizkiy, A. Tsymbal, S. Puuronen, and D. Patterson, "Feature extraction for dynamic integration of classifiers," *Fundamenta Informaticae*, vol. 77, no. 3, pp. 243–275, Aug. 2007, ISSN: 0169-2968.

[39] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *ACM Transactions on Knowledge Discovery from Data*, vol. 3, no. 1, pp. 1–57, 2009.

[40] M. Radovanovic, A. Nanopoulos, and M. Ivanovic, "Nearest neighbors in high-dimensional data: The emergence and influence of hubs," in *Proceedings of the 26th International Conference On Machine Learning, ICML 2009*, vol. 382, Jan. 2009, p. 109. DOI: 10.1145/1553374.1553485.

[41] P. Ray, S. S. Reddy, and T. Banerjee, "Various dimension reduction techniques for high dimensional data analysis: A review," *Artificial Intelligence Review*, vol. 54, no. 5, pp. 3473–3515, 2021.

[42] A. Maciel-Guerra, G. P. Figueredo, F. J. V. Zuben, E. Marti, J. Twycross, and M. J. C. Alcocer, "Subspace-based dynamic selection: A proof of concept using protein microarray data," in *WCCI - World Congress on Computational Intelligence, The International Joint Conference on Neural Networks (IJCNN) 2020, Glasgow, UK, July 19-24, 2020*, IEEE, 2020. DOI: 10.1109/IJCNN48605.2020.9207611.

[43] A. Maciel-Guerra, G. P. Figueredo, and J. Twycross, "Classifier subspace-based dynamic selection for high-dimensional data," *In Review*, 2022.

[44] M. P. Ponti Jr, "Combining classifiers: From the creation of ensembles to the decision fusion," in *2011 24th SIBGRAPI Conference on Graphics, Patterns, and Images Tutorials*, IEEE, 2011, pp. 1–10.

[45] A. T. Sergio, T. P. de Lima, and T. B. Ludermir, "Dynamic selection of forecast combiners," *Neurocomput.*, vol. 218, no. C, pp. 37–50, Dec. 2016, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2016.08.072.

[46] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen, "Dynamic integration of classifiers for handling concept drift," *Inf. Fusion*, vol. 9, no. 1, pp. 56–68, Jan. 2008, ISSN: 1566-2535. DOI: 10.1016/j.inffus.2006.11.002.

[47] J. Xiao, L. Xie, C. He, and X. Jiang, "Dynamic classifier ensemble model for customer classification with imbalanced class distribution," *Expert Syst. Appl.*, vol. 39, no. 3, pp. 3668–3675, Feb. 2012, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2011.09.059.

[48] J. Xiao, C. He, X. Jiang, and D. Liu, "A dynamic classifier ensemble selection approach for noise data," *Inf. Sci.*, vol. 180, no. 18, pp. 3402–3421, Sep. 2010, ISSN: 0020-0255. DOI: 10.1016/j.ins.2010.05.021.

[49] T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 1, pp. 66–75, Jan. 1994, ISSN: 0162-8828. DOI: 10.1109/34.273716.

[50] R. Mousavi and M. Eftekhari, "A new ensemble learning methodology based on hybridization of classifier ensemble selection approaches," *Appl. Soft Comput.*, vol. 37, no. C, pp. 652–666, Dec. 2015, ISSN: 1568-4946. DOI: 10.1016/j.asoc.2015.09.009.

[51] R. M. Cruz, "Dynamic selection of ensemble of classifiers using meta-learning," PhD thesis, École de Technologie Supérieure, Université du Québec, Montreal, Canada, Jun. 2016.

[52] D. Ruta and B. Gabrys, "Classifier selection for majority voting," *Information Fusion*, vol. 6, no. 1, pp. 63–81, 2005, Diversity in Multiple Classifier Systems, ISSN: 1566-2535. DOI: 10.1016/j.inffus.2004.04.008.

[53] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[54] J. Moody and C. J. Darken, "Fast learning in networks of locally tuned processing units," *Neural Computing*, vol. 1, pp. 281–294, 1989.

[55] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington DC, 1961.

[56] S. Maldonado and R. Weber, "A wrapper method for feature selection using support vector machines," *Information Sciences*, vol. 179, pp. 2208–2217, 2009. DOI: 10.1016/j.ins.2009.02.014.

[57] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[58] M. Sabourin, A. Mitiche, D. Thomas, and G. Nagy, "Classifier combination for hand-printed digit recognition," in *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*, Oct. 1993, pp. 163–166. DOI: 10.1109/ICDAR.1993.395758.

[59] K. Woods, W. P. Kegelmeyer, and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 405–410, Apr. 1997, ISSN: 0162-8828. DOI: 10.1109/34.588027.

[60] G. Giacinto and F. Roli, "Methods for dynamic classifier selection," in *Proceedings 10th International Conference on Image Analysis and Processing*, 1999, pp. 659–664. DOI: 10.1109/ICIAP.1999.797670.

[61] G. Giacinto and F. Roli, "Dynamic classifier selection based on multiple classifier behaviour," *Pattern Recognition*, vol. 34, Nov. 2002.

[62] P. C. Smits, "Multiple classifier systems for supervised remote sensing image classification based on dynamic classifier selection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 4, pp. 801–813, Apr. 2002, ISSN: 0196-2892. DOI: 10.1109/TGRS.2002.1006354.

[63] 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*, 2006, pp. 1310–1316. DOI: 10.1109/IJCNN.2006.246844.

[64] M. C. P. de Souto, R. G. F. Soares, A. Santana, and A. M. P. Canuto, "Empirical comparison of dynamic classifier selection methods based on diversity and accuracy for building ensembles," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, Jun. 2008, pp. 1480–1487. DOI: 10.1109/IJCNN.2008.4633992.

[65] T. Woloszynski and M. Kurzynski, "On a new measure of classifier competence applied to the design of multiclassifier systems," in *Proceedings of the 15th International Conference on Image Analysis and Processing*, ser. ICIAP '09, Vietri sul Mare, Italy: Springer-Verlag, 2009, pp. 995–1004, ISBN: 978-3-642-04145-7. DOI: 10.1007/978-3-642-04146-4_106.

[66] B. Antosik and M. Kurzynski, “New measures of classifier competence - heuristics and application to the design of multiple classifier systems,” in *Computer Recognition Systems 4*, R. Burduk, M. Kurzyński, M. Woźniak, and A. Żołnierek, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 197–206, ISBN: 978-3-642-20320-6.

[67] T. Woloszynski, M. Kurzynski, P. Podsiadlo, and G. W. Stachowiak, “A measure of competence based on random classification for dynamic ensemble selection,” *Information Fusion*, vol. 13, no. 3, pp. 207–213, 2012, ISSN: 1566-2535. DOI: 10.1016/j.inffus.2011.03.007.

[68] P. R. Cavalin, R. Sabourin, and C. Y. Suen, “Dynamic selection approaches for multiple classifier systems,” *Neural Computing and Applications*, vol. 22, no. 3, pp. 673–688, Mar. 2013, ISSN: 1433-3058. DOI: 10.1007/s00521-011-0737-9.

[69] A. L. Brun, A. S. Britto, L. S. Oliveira, F. Enembreck, and R. Sabourin, “Contribution of data complexity features on dynamic classifier selection,” in *2016 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2016, pp. 4396–4403. DOI: 10.1109/IJCNN.2016.7727774.

[70] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997, ISSN: 0022-0000. DOI: 10.1006/jcss.1997.1504.

[71] R. E. Schapire, “The strength of weak learnability,” *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[72] T. K. Ho, “The random subspace method for constructing decision forests,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[73] Ş. Y. Sağlam and W. N. Street, “Distant diversity in dynamic class prediction,” *Annals of Operations Research*, vol. 263, no. 1, pp. 5–19, Apr. 2018. DOI: 10.1007/s10479-016-2328-8.

[74] R. M. Cruz, R. Sabourin, and G. D. Cavalcanti, "Meta-des.oracle," *Inf. Fusion*, vol. 38, no. C, pp. 84–103, Nov. 2017, ISSN: 1566-2535. DOI: 10.1016/j.inffus.2017.02.010.

[75] B. Ghojogh, M. N. Samad, S. A. Mashhadi, T. Kapoor, W. Ali, F. Karray, and M. Crowley, "Feature selection and feature extraction in pattern analysis: A literature review," *ArXiv*, vol. abs/1905.02845, 2019.

[76] Z. Hira and D. Gillies, "A review of feature selection and feature extraction methods applied on microarray data," *Advances in Bioinformatics*, vol. 2015, pp. 1–13, Jul. 2015. DOI: 10.1155/2015/198363.

[77] A. Bommert, X. Sun, B. Bischl, J. Rahnenführer, and M. Lang, "Benchmark for filter methods for feature selection in high-dimensional classification data," *Computational Statistics & Data Analysis*, vol. 143, p. 106 839, 2020, ISSN: 0167-9473. DOI: 10.1016/j.csda.2019.106839.

[78] T. Marill and D. M. Green, "On the effectiveness of receptors in recognition systems," *IEEE Transactions on Information Theory*, vol. 9, pp. 11–17, 1973. DOI: 10.1371/journal.pone.0130814.

[79] H. Liu and H. Motoda, *Feature extraction, construction and selection. A data mining perspective*, ser. The Springer International Series in Engineering and Computer Science. Springer US, 1978. DOI: 10.1007/978-1-4615-5725-8.

[80] E. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[81] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006. DOI: 10.1016/j.neucom.2005.12.126.

[82] A. C. P. Kulaif and F. J. V. Zuben, "Improved regularization in extreme learning machines," *In: 11th Brazilian Congress on Computational Intelligence, 2013, Ipojuca – PE. Proceedings of the 11th Brazilian Congress on Computational Intelligence*, vol. 1, pp. 1–6, 2013.

[83] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012. DOI: 10.1109/TSMCB.2011.2168604.

[84] H. M. Zawbaa, E. Emary, C. Grosan, and V. Snasel, "Large-dimensionality small-instance set feature selection: A hybrid bio-inspired heuristic approach," *Swarm and Evolutionary Computation*, vol. 42, pp. 29–42, 2018, ISSN: 2210-6502. DOI: 10.1016/j.swevo.2018.02.021.

[85] L. I. Kuncheva and J. J. Rodríguez, "On feature selection protocols for very low-sample-size data," *Pattern Recognition*, vol. 81, pp. 660–673, 2018, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2018.03.012.

[86] E. Muller, S. Gunnemann, I. Assent, and T. Seidl, "Evaluating clustering in subspace projections of high dimensional data," *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 1270–1281, Aug. 2009, ISSN: 2150-8097. DOI: 10.14778/1687627.1687770.

[87] J. Tian and M. Gu, "Subspace clustering based on self-organizing map," in *Proceeding of the 24th International Conference on Industrial Engineering and Engineering Management 2018*, G. Q. Huang, C.-F. Chien, and R. Dou, Eds., Singapore: Springer Singapore, 2019, pp. 151–159. DOI: 10.1007/978-981-13-3402-3_17.

[88] R. Basri, T. Hassner, and L. Zelnik-Manor, "Approximate nearest subspace search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 2, pp. 266–278, Feb. 2011, ISSN: 0162-8828. DOI: 10.1109/TPAMI.2010.110.

[89] M. Hund, M. Behrisch, I. Färber, M. Sedlmair, T. Schreck, T. Seidl, and D. Keim, "Subspace nearest neighbor search - problem statement, approaches, and discussion," in *Similarity Search and Applications*, G. Amato, R. Connor, F. Falchi, and C. Gennaro, Eds., Cham: Springer International Publishing, 2015, pp. 307–313, ISBN: 978-3-319-25087-8.

[90] X.-S. Yang, S. Lee, S. Lee, and N. Theera-Umpon, "Information Analysis of High-Dimensional Data and Applications," *Mathematical Problems in Engineering*, vol. 2015,

no. ii, pp. 1–2, 2015, ISSN: 1024-123X. DOI: 10.1155/2015/126740. [Online]. Available: http://www.hindawi.com/journals/mpe/2015/126740/.

[91] N. Tomasev, M. Radovanovic, D. Mladenic, and M. Ivanovic, "The role of hubness in clustering high-dimensional data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 3, pp. 739–751, Mar. 2014, ISSN: 1041-4347. DOI: 10.1109/TKDE.2013.25.

[92] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, *Feature selection datasets at arizona state university*, http://featureselection.asu.edu/datasets.php, Accessed: August, 2018.

[93] D. Dheeru and E. Karra Taniskidou, *UCI machine learning repository*, 2017. [Online]. Available: http://archive.ics.uci.edu/ml.

[94] G. V. Trunk, "A problem of dimensionality: A simple example," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 3, pp. 306–307, Jul. 1979, ISSN: 0162-8828. DOI: 10.1109/TPAMI.1979.4766926.

[95] 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*, vol. 12, pp. 2825–2830, 2011.

[96] R. M. O. Cruz, L. G. Hafemann, R. Sabourin, and G. D. C. Cavalcanti, "DESlib: A Dynamic ensemble selection library in Python," *arXiv preprint arXiv:1802.04967*, 2018.

[97] R. L. Iman and J. M. Davenport, "Approximations of the critical region of the fbietkan statistic," *Communications in Statistics - Theory and Methods*, vol. 9, no. 6, pp. 571–595, 1980. DOI: 10.1080/03610928008827904.

[98] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006, ISSN: 1532-4435.

[99] Z. Yu, Z. Wang, J. You, J. Zhang, J. Liu, H. Wong, and G. Han, “A new kind of nonparametric test for statistical comparison of multiple classifiers over multiple datasets,” *IEEE Transactions on Cybernetics*, vol. 47, no. 12, pp. 4418–4431, Dec. 2017, ISSN: 2168-2267. DOI: 10.1109/TCYB.2016.2611020.

[100] D. R. Cox, “The regression analysis of binary sequences,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 20, no. 2, pp. 215–242, 1958.

[101] Y. Freund and R. Schapire, “A decision theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[102] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, third. Morgan Kaufmann Publishers, 2011.

[103] G. J. McLachlan, *Discriminant analysis and statistical pattern recognition.* Wiley, New York, 2004.

[104] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning.* Springer, New York, 2003.

[105] F. Benoit, M. van Heeswijk, Y. Miche, M. Verleysen, and A. Lendasse, “Feature selection for nonlinear models with extreme learning machines,” *Neurocomputing*, vol. 102, pp. 111–124, 2013.

[106] E. Marti, X. Wang, N. Jambari, C. Rhyner, J. Olzhausen, J. J. Pérez-Barea, G. P. Figueredo, and M. J. C. Alcocer, “Novel in vitro diagnosis of equine allergies using a protein array and mathematical modelling approach: A proof concept using insect bite hypersensitivity,” *Veterinary Immunology and Immunopathology*, vol. 167, pp. 171–177, 2015.

[107] K. A. Vigh-Conrad, D. F. Conrad, and D. Preuss, “A protein allergen microarray detects specific ige to pollen surface, cytoplasmic, and commercial allergen extracts,” *PLOS ONE*, vol. 5, no. 4, pp. 1–11, Apr. 2010. DOI: 10.1371/journal.pone.0010174.

[108] P. Zhu, W. Zhu, Q. Hu, C. Zhang, and W. Zuo, "Subspace clustering guided unsupervised feature selection," *Pattern Recognition*, vol. 66, pp. 364–374, 2017, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2017.01.016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320317300158.

[109] H. D. Gangurde, "Feature selection using clustering approach for big data," in *International Journal of Computer Applications*, 2014.

[110] I. Guyon, "Design of experiments for the nips 2003 variable selection benchmark," 2003.

[111] G. F. Jenks, "The data model concept in statistical mapping," in *International Yearbook of Cartography*, vol. 7, 1967, pp. 186–190.

[112] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: A corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, 2010.

[113] S. A. Dudani, "The distance-weighted k-nearest-neighbor rule," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, no. 4, pp. 325–327, Apr. 1976, ISSN: 0018-9472. DOI: 10.1109/TSMC.1976.5408784.

[114] J. E. S. Macleod, A. Luk, and D. M. Titterington, "A re-examination of the distance-weighted k-nearest neighbor classification rule," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 17, no. 4, pp. 689–696, Jul. 1987, ISSN: 0018-9472. DOI: 10.1109/TSMC.1987.289362.

[115] R. M. O. Cruz, R. Sabourin, and G. D. C. Cavalcanti, "Meta-des.h: A dynamic ensemble selection technique using meta-learning and a dynamic weighting approach," in *2015 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2015, pp. 1–8. DOI: 10.1109/IJCNN.2015.7280594.

[116] K. Stapor, "Evaluating and comparing classifiers: Review, some recommendations and limitations," in *Proceedings of the 10th International Conference on Computer Recognition Systems CORES 2017*, M. Kurzynski, M. Wozniak, and R. Burduk,

Eds., Cham: Springer International Publishing, 2018, pp. 12–21, ISBN: 978-3-319-59162-9.

[117] G. I. Webb, “Multiboosting: A technique for combining boosting and wagging,” *Machine Learning*, vol. 40, no. 2, pp. 159–196, Aug. 2000, ISSN: 1573-0565. DOI: 10.1023/A:1007659514849.

[118] S. Mishra and M. Osman. (2018). Nonparametric statistics, [Online]. Available: www.ce.memphis.edu/7012/L10_Nonparametric%20Student%20Notes%20(Updated).pdf (visited on 05/19/2018).

[119] D. I. Foreman and G. W. Corder, “Critical value tables,” in *Nonparametric Statistics: A Step-by-Step Approach*, Wiley, May 2014, ch. Appendix B, ISBN: 978-1-118-84031-3.

[120] T. Löfström, “On effectively creating ensembles of classifiers,” PhD thesis, Department of Computer Systems Sciences, Stockholm University, Stockholm, Sweden, 2015, ISBN: 978-91-7649-179-9.

[121] S. Garcia, A. Fernandez, J. Luengo, and F. Herrera, “Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power,” *Information Science*, vol. 180, no. 10, pp. 2044–2064, May 2010, ISSN: 0020-0255. DOI: 10.1016/j.ins.2009.12.010.

[122] F. Carvalho and C. Zanchettin. (2018). Aprendizagem de máquina - avaliação de modelos - nemenyi critical, [Online]. Available: http://www.cin.ufpe.br/~fatc/AM/Nemenyi_critval.pdf (visited on 05/20/2018).

[123] C. J. Wild and G. A. F. Seber, “The wilcoxon rank-sum test,” in *Chance encounters: A first course in data analysis and inference*, Wiley, 1999, ch. 10, ISBN: 9780471329367.

[124] M. Tomczak and E. Tomcak, “The need to report effect size estimates revisited. an overview of some recommended measures of effect size,” in *Trends in Sport Sciences*, vol. 1, Jan. 2014, pp. 19–25.

[125] J. Cohen, "The significance of a product moment," in *Statistical Power Analysis for the Behavioral Sciences*, Lawrence Erlbaum Associates, 1988, ch. 3, ISBN: 0-8058-0283-5.