Artificial Intelligence: Foundations, Theory, and Algorithms

Verónica Bolón-Canedo
Noelia Sánchez-Maroño
Amparo Alonso-Betanzos

Feature Selection for High-Dimensional Data

Springer

Verónica Bolón-Canedo • Noelia Sánchez-Maroño
Amparo Alonso-Betanzos

Feature Selection for High-Dimensional Data

Springer

Verónica Bolón-Canedo
Facultad de Informática
Universidad de A Coruña
A Coruña, Spain

Noelia Sánchez-Maroño
Facultad de Informática
Universidad de A Coruña
A Coruña, Spain

Amparo Alonso-Betanzos
Facultad de Informática
Universidad de A Coruña
A Coruña, Spain

ISSN 2365-3051 ISSN 2365-306X (electronic)
Artificial Intelligence: Foundations, Theory, and Algorithms
ISBN 978-3-319-21857-1 ISBN 978-3-319-21858-8 (eBook)
DOI 10.1007/978-3-319-21858-8

Library of Congress Control Number: 2015951087

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

To our families

Foreword

The topic of variable selection in high-dimensional spaces (often with hundreds or thousands of dimensions) has attracted considerable attention in data mining research in previous years, and it is common in many real problems.

In a nutshell, feature selection is a process that chooses an optimal subset of features according to a certain criterion. The selection of the criterion must be done according to the purpose of feature selection, usually with the aim of improving the prediction accuracy of the data mining algorithm used to learn a model. Generally, the objective is to identify the features in the dataset which are important and discard others as redundant or irrelevant. The problem is especially relevant when we are managing a huge number of features and the learning algorithm loses prediction capacity using all of them. Since feature selection reduces the dimensionality of the data, the data mining algorithms can run faster and obtain better outcomes by using feature selection.

The publication of the book "Feature Selection for High-Dimensional Data" written by Verónica Bolón-Canedo, Noelia Sánchez-Maroño and Amparo Alonso-Betanzos is an important event. The book offers a coherent and comprehensive approach to feature subset selection in the scope of classification problems.

We can shortly outline the three parts found when reading the book: foundations, real application problems and challenges. First, the authors focus on the analysis and synthesis of feature selection algorithms, presenting a comprehensive review of basic concepts and experimental results of the most well known algorithms. Second, an interesting novelty and contribution of the book is how it addresses different real scenarios with high-dimensional data, showing the use of feature selection algorithms in different contexts with different requirements and information: microarray data, intrusion detection, tear film lipid layer classification and cost based features. Third, the book also delves into the scenario of big dimension, sometimes combined with massive amounts of data as big data. It pays attention to important problems under high-dimensional spaces: scalability, distributed processing and real-time processing. These are scenarios that open up new and interesting challenges for researchers.

This triple orientation makes it a different and original book, written in a very clear and comprehensive style. This book allows readers to delve into feature selection from both the theoretical and practical perspective. Furthermore, it shows the major challenges in this well established field, which in turn is also a very dynamic one because of its great importance in data preprocessing.

The book is authored by great experts in the field, who make an important contribution to the topic. It is a must read for anyone who is concerned with the design or application of data mining algorithms for getting knowledge from high-dimensional data.

Granada, Spain
March 2015

Francisco Herrera

Preface

Feature selection (FS) has been embraced as one of the high activity research areas during the last few years, due to the appearance of datasets containing hundreds of thousands of features (variables). Thus, feature selection was employed to be able to better model the underlying process of data generation, and to reduce the cost of acquiring the variables. Furthermore, from the Machine Learning algorithms viewpoint, as FS is able to reduce the dimensionality of the problem, it can be used for maintaining and, most of the time, improving the algorithms' performance, while reducing computational costs. Nowadays, the advent of Big Data has brought unprecedented challenges to machine learning researchers, who now have to deal with huge volumes of data, in terms of both instances and features, making the learning task more complex and computationally demanding. Specifically, when dealing with an extremely large number of input features, learning algorithms' performance can degenerate due to overfitting, learned models decrease their interpretability as they become more complex, and finally speed and efficiency of the algorithms decline in accordance with size.

A vast body of feature selection methods exist in the literature, including filters based on distinct metrics (e.g., entropy, probability distributions or information theory) and embedded and wrapper methods using different induction algorithms. The proliferation of feature selection algorithms, however, has not brought about a general methodology that allows for intelligent selection from existing algorithms. In order to make a correct choice, a user not only needs to know the domain well, but also is expected to understand the technical details of available algorithms. On top of this, most algorithms were developed when dataset sizes were much smaller, but, nowadays, distinct trade-offs are required for the case of small-scale and large-scale learning problems. Small-scale learning problems are subject to the usual approximation-estimation trade-off. In the case of large-scale learning problems, the trade-off is more complex because it involves not only the accuracy of the selection but also other aspects, such as stability (i.e., the sensitivity of the results to training set variations) or scalability. All these aspects are addressed in the first four chapters of the book, to give the reader not only a broad perspective of the state of the art, but also a critical review of the behavior of the methods in different situations, such as noise, correlation of the variables, redundancy, etc. In this way, the advanced reader and the researcher can have a reference against which to compare the results of new methods.

Also, the book addresses the "big" feature dimensionality factor of the data, which has received much less attention than the "Big Instance Size" factor of the data (devoted to the large number of instances). It is intended to be a comprehensive book completely devoted to analyzing the evolution of feature dimensionality in the last few decades, the state-of-the-art feature selection methods and the emerging challenges in this field. This aspect is emphasized in Chapter 4, devoted to the microarray datasets, a challenge for feature selection due to their small sample size (in the order of hundreds), but large number of variables (in the order of thousands). Chapter 5, on the other hand, explores two real applications that situate the reader in the contexts related with imbalanced datasets, the importance of the cost of selecting features, etc.

Finally, Chapter 6 is devoted to the emerging challenges of the discipline. Feature selection is often regarded as one of the most important tasks in the omnipresent actual scenario of Big Data research, since the emersion of high-dimensionality requires feature selection strategies that are capable of coping with the explosion of features. As an example, state-of-the-art feature selection methods such as minimum Redundancy Maximum Relevance (mRMR) or Support Vector Machine based on Recursive Feature Elimination (SVM-RFE) take more than a day of computational effort to deal with datasets composed of 0.5 million features. Thus, an exhaustive and updated background in the topic could be very useful for the data analytics and computational intelligence communities since it underlines the emerging trend of high-dimensionality and deliberates on how the existing methods are prepared to face the arising challenges. This book invites readers to explore the different issues associated with feature selection for high-dimensional data:

- The evolution of feature dimensionality in the last few decades.
- State-of-the-art feature selection methods.
- Adequacy of feature selection methods to solve real problems.
- Emerging challenges for feature selection.

The target audience of this book comprises anyone interested in improving their understanding of feature selection. Researchers could take advantage of this extensive review on state-of-the-art feature selection methods and gather new ideas from the emerging challenges described. Practitioners in industry should find new directions and opportunities from the topics covered. Finally, we hope our readers enjoy reading this book as much as we enjoyed the adventure of its production.

We are thankful to all our collaborators, who helped with some of the research involved in this book. We would also like to acknowledge our families and friends for their invaluable support, and not only during this writing process. Last but not least, we are indebted and grateful to Francisco Herrera, from the University of Granada, who encouraged us to compile this book.

A Coruña, Spain
March 2015

Verónica Bolón-Canedo
Noelia Sánchez-Maroño
Amparo Alonso-Betanzos

Contents

Chapter 1
Introduction to High-Dimensionality

Abstract In an era in which the complexity and volume of data available in the field of machine learning is growing daily, feature selection plays an important role, helping to reduce the "high-dimensionality" of some problems. In this chapter, the problematics and characteristics of these "high-dimensional" datasets will be presented. Section 1.1 introduces the need for feature selection from the advent of Big Data. In Section 1.2, we outline the main applications that are promoting feature selection. Then, in Section 1.3, the inherent characteristics of some problems that may hinder the feature selection process are also discussed. Finally, we give an overview of the different chapters of this book in Section 1.4.

In an era in which the volume and complexity of datasets is continuously growing, machine learning techniques have become indispensable when extracting useful information from huge amounts of meaningless data. Among the many techniques of machine learning, feature selection is characterized by electing the attributes that allow a problem to be clearly defined, apart from those that are irrelevant or redundant. There are numerous feature selection methods that, traditionally, have been categorized into three groups: filters, wrappers and embedded. In addition, new methods combining existing methods or other machine learning techniques are continuously appearing to deal with the challenges of today's datasets.

This book presents an overview of the most popular feature selection methods, analyzing their performance under different conditions. Then, their application to real problems is presented, paying particular attention to one of the problems that has aroused significant interest in the scientific community: microarray datasets. Furthermore, emerging challenges will be propounded.

The purpose of this introductory chapter is to briefly present some real problems that generate lots of data, mainly in the feature dimension, turning to huge datasets where dimensionality reduction becomes a necessity. Subsequently, this chapter focuses on setting out some inherent difficulties that these datasets may have and, therefore, representing a challenge for any learning technique, including feature se-

V. Bolón-Canedo et al., *Feature Selection for High-Dimensional Data*, Artificial Intelligence: Foundations, Theory, and Algorithms, DOI 10.1007/978-3-319-21858-8_1

lection. The last section is devoted to guiding the reader through the different chapters of this book.

1.1 The Need for Feature Selection

In recent years, most enterprises and organizations have stored large amounts of data in a systematic way, without any clear potential use. In addition, with the rapid growth of the Internet, lots of data come in different formats—text, multimedia, etc.—from many different sources—systems, sensors, mobile devices, etc.—and they require new tools in order to be analyzed and processed so as to enable the extraction of useful information. Most of these data have been generated in the last few years while we continue to create a quintillion bytes daily [1]. Therefore, big data of large volume and ultrahigh dimensionality have emerged in various machine learning applications, such as text mining and information retrieval [2]. For instance, Weinberger et al. [3] have studied a collaborative email/spam filtering task with 16 trillion unique features. These datasets have brought an interesting challenge to the research community, and, citing [4], as researchers, "our task is to find a needle in a haystack, teasing the relevant information out of a vast pile of glut."

The ultrahigh dimensionality not only incurs unbearable memory requirements and high computational cost in training, but also deteriorates the generalization ability because of the "curse of dimensionality" issue. According to [4], the colorful phrase the "curse of dimensionality" was apparently coined by Richard Bellman in [5], in connection with the difficulty of optimization by exhaustive enumeration on product spaces. This phenomena arises when analyzing and organizing data in high-dimensional spaces that do not occur in low-dimensional settings. Any dataset is usually represented by a matrix where the rows are the instances or samples that have been recorded and the columns are the attributes or features that are required to represent the problem at hand. Therefore, to tackle the "curse of dimensionality" problem, the dataset can be summarized by finding "narrower" matrices that in some sense are close to the original. These narrow matrices have only a small number of samples and/or a small number of attributes, and therefore can be used much more efficiently than the original large matrix can. The process of finding these narrow matrices is called *dimensionality reduction*.

Among the dimensionality reduction techniques, feature extraction addresses the problem of finding the most compact and informative set of features of a given problem, to improve the efficiency or data storage or processing. The problem of feature extraction is decomposed into two steps: feature construction and feature selection. Feature construction methods complement the human expertise to convert "raw" data into a set of useful features. It may be considered a preprocessing transformation that may include standardization, normalization, discretization, signal enhancement, extraction of local features, etc. Some construction methods do not alter the space dimensionality, while others enlarge it, reduce it or can act in either direction. But one should be aware of not losing information at the feature construction stage.

Guyon and Elisseeff [6] argued that it is always better to err on the side of being too inclusive rather than risking discarding useful information. Adding many features seems reasonable but it comes at a price: it increases the dimensionality of the patterns and thereby immerses the relevant information into a sea of possibly irrelevant, noisy or redundant features. Feature selection is the process in which the number of initial features is reduced and a subset of those that retain enough information for obtaining good, or even better, performance results is selected.

Given the growing number of high-dimensional datasets, feature selection has gained increasing interest in the field of machine learning, addressing many of its disciplines, such as clustering [7, 8], regression [9, 10] and classification [11, 12], in both supervised and unsupervised ways. This book would be too long if all of them were to be addressed, which is why we have focused on the application of feature selection in supervised classification problems.

1.2 When Features Are Born

Feature selection research dates back to the 1960s [13]. Hughes [14] used a general parametric model to study the accuracy of a Bayesian classifier as a function of the number of features. He concludes: "measurement selection, reduction and combination are not proposed as developed techniques. Rather, they are illustrative of a framework for further investigation." Since then, the research in feature selection has been a challenging field despite the skepticism of some researchers, for example, in a discussion of Miller [15], R.L. Plackett stated: "If variable elimination has not been sorted out after two decades of work assisted by high-speed computing, then perhaps the time has come to move on to other problems." Nowadays, it is well known that feature selection plays a crucial role in some real problems to reduce their dimensionality, and there are numerous books and papers regarding this issue and their number is continuously growing [16, 17, 18].

There are many ways to solve problems, and just as many ways to describe them. Thus, the means of generating features will be markedly different due to the different representations that are of interest. Therefore, the myriad of feature types and their mixtures are becoming a norm in many of today's real applications and, together with the rapid advancements in computing and information technologies, they are major contributors to feature explosion [1]. The next few paragraphs outline the main applications that are promoting feature selection:

- Computational biology. Bioinformatics tools have been widely applied to genomics, proteomics, gene networks, structure prediction, disease diagnosis and drug design. DNA microarrays have been extensively used in simultaneously monitoring mRNA expressions of thousands of genes in many areas of biomedical research [19]. These datasets typically consist of several hundred samples, as opposed to thousands of genes, whereby the role of the feature selection is crucial.

- Image classification. This field has become an interesting research field by analyzing the numerical properties of various image features and organizing data into categories. All the advances in image capturing, storage, and Internet technologies have made vast amounts of image data available. Since then, a huge variety of images is becoming accessible to the public, from photo collections to web pages, or even video databases. Since visual media need large amounts of memory and computing power for processing and storage, there is a requirement for efficiently indexing and retrieving visual information from an image database.
- Financial engineering and risk management. Technological revolution and trade globalization have introduced a new era of financial markets. Over the last three decades, an enormous number of new financial products have been created to meet customer demands. The stock market trend is very complex and is influenced by various factors. Therefore it is very necessary to find out the most significant factors of the stock market and feature selection can be applied to achieve this goal.

Other problems include handwritten digit recognition, structure patterns, health studies, survival analysis, etc.

1.3 Intrinsic Characteristics of Data

As discussed in the previous section, there are numerous problems that can benefit from the application of feature selection techniques. Real datasets are acquired through a variety of methods. Some methods of acquisition can be very expensive (personal or/and material), hindering the number of samples collected; other methods may add noise to the data gathered; and so on. Therefore, data may end up with inherent difficulties. This section focuses on the characteristics that data may present, thereby hampering the feature selection stage and the subsequent classification.

1.3.1 Small Sample Size

The *small sample size problem* is when "only m samples are available in an n-dimensional vector space with $m < n$," citing Fukunaga [20]. This author says that this problem is often encountered in pattern recognition, particularly when n is very large. Nowadays, high dimension, low sample size (HDLSS) data are becoming increasingly common in various fields. These include the genetic microarrays previously presented, chemometrics, medical imaging, text recognition, face recognition, finance, and so on. Many of the features of these problems do not facilitate an adequate classification; rather they hinder it. To avoid this problem, the importance of feature selection has been stressed and several methods of feature selection for HDLSS data have been proposed in the literature, for example, the nearest shrunken centroids method [21], feature annealed independence rules [22], minimum redundancy maximum relevance feature selection [23], and so on. A comparative study of some basic feature-selection methods in settings involving thousands of features can be found in [24].

1.3.2 Class Imbalance

Data are said to suffer the *Class Imbalance Problem* when the class distributions are highly imbalanced. This occurs when a dataset is dominated by a major class or by classes which have significantly more instances than the other rare or minority classes in the data. For the two-class case, without loss of generality, one assumes that the minority or rare class is the positive class, and the majority class is the negative class. Often the minority class is very infrequent, such as 1% of the dataset. If one applies the most traditional classifiers on the dataset, they are likely to predict everything as negative (the majority class) [25]. However, typically, people have more interest in learning about rare classes. For example, applications such as medical diagnosis prediction of rare but important disease, such as cancer, where it is common to have fewer cancer patients than healthy patients. Similar situations are observed in other areas, such as detecting fraud in banking operations, detecting network intrusions, managing risk, predicting technical equipment failure, e-mail foldering and face recognition [26].

Therefore, in the last few years, many solutions have been proposed to deal with this problem [27, 28, 29], which can be categorized into three major groups [30]:

1. Data sampling, in which the training samples are modified in such a way as to produce a more or less balanced class distribution that allows classifiers to improve their performance. Traditional data sampling techniques are undersampling methods, which create a subset of the original dataset by eliminating instances; oversampling methods, which create a superset of the original dataset by replicating some instances or creating new instances from existing ones; and, finally, hybrid methods that combine both sampling methods. One of the most employed resampling techniques is the so-called SMOTE [31], where the minority class is oversampled by taking each minority class sample and introducing synthetic examples along the line segments joining any or all of the k minority class nearest neighbors.
2. Algorithmic modification: This procedure is oriented toward the adaptation of base learning methods to be more attuned to class imbalance issues. For example, Cieslak and Chawla [32] consider the Hellinger distance [33] as a decision tree splitting criterion, which proved to be skew-insensitive, i.e., it captures the divergence in distributions without being dominated by the class priors.
3. Cost-sensitive learning: This type of solution incorporates approaches at the data level, at the algorithmic level, or at both levels combined, considering higher costs for the misclassification of examples of the positive class with respect to the negative class, and therefore trying to minimize higher cost er-

rors. As said previously, when dealing with imbalanced problems it is usually more important to recognize the positive instances rather than the negative ones. Therefore, the cost of misclassifying a positive instance is higher than the cost of misclassifying a negative one. As a classical example, the reader may refer to a diagnosis problem in which it is often less dangerous to obtain a false positive than a false negative [34].

1.3.3 Data Complexity

According to [35], difficulties in classification can be traced to three sources:

1. Class ambiguity refers to a situation where there are cases in a classification problem whose classes cannot be distinguished by any classification algorithm using the given features. Classes can be ambiguous for two reasons: a) the class concepts are poorly defined and, so, intrinsically inseparable, and b) the features chosen to represent the different samples are not sufficient for indicating differences between instances. An example of the former would be the lowercase letter "l" and the numeral "1", which are the same in many fonts.
2. Boundary complexity: a class boundary is complex if it requires a long description, possibly including a listing of all the points together with their class labels. This aspect of difficulty comes from the nature of the problem, and is unrelated to the sampling process. It exists even if a complete sample is given and if the classes are well defined.
3. Sample sparsity: small sample size and dimensionality-induced sparsity introduce another layer of difficulty through a lack of constraints of the generalization rules.

In real-world situations, a problem often becomes difficult because of a mixture of these three effects. Sampling density is more critical for an intrinsically complex problem than an intrinsically simple problem (e.g., a linearly separable problem with wide margins). If the sample is too sparse, an intrinsically complex problem may appear deceptively simple [36].

Data complexity measures are a recent proposal to represent characteristics of the data which are considered difficult in classification tasks according to the previously introduced categorization [36]. The different types of measures can be grouped as follows [35, 36]:

- Measures of class overlapping. These measures focus on the effectiveness of a single feature dimension in separating the classes, or the composite effects of a number of dimensions. These measures are F1 (*maximum Fisher's discriminant ratio*), F2 (*volume of overlapping ratio*) and F3 (*maximum (individual) feature efficiency*).
- Measures of separability of classes. These measures evaluate to what extent two classes are separable by examining the existence and shape of the class boundary. The contributions of individual feature dimensions are combined and sum-

marized in a single score, usually a distance metric, rather than evaluated separately. Linear separability measures, such as L1 and L2, which have been intensively discussed in the literature are included here, besides mixture identifiability measures (N1, N2 and N3); see [35, 36] for details.
- Measures of geometry, topology, and density of manifolds. These measures give *indirect* characterizations of class separability. They assume that a class is made up of a single or multiple manifolds that form the support of probability distribution of the given class. The shape, position, and interconnectedness of these manifolds give hints on how well two classes are separated, but they do not describe separability by design. In this group, measures of nonlinearity of a linear classifier (L3) or of a nearest neighbour classifier (N4) are found. Specifically, the measure that determines the space covered by ε-neighbourhoods (T1) or the ratio of instances/features (T2).

1.3.4 Dataset Shift

Many repositories (see Appendix A, Section A.2.1) provide datasets that are already divided into training and test sets. Sometimes, these datasets come from competitions. Herein, it is usual to provide a test set slightly different from the training data to make the competition task more realistic. For example, the KDD Cup 99 competition task [37] was to build a network intrusion detector, a predictive model capable of distinguishing between intrusions or attacks and "good" normal connections. The test set provided at this competition included specific attack types that were not in the training data. Later, this division was maintained to facilitate further comparative studies with the results of the competition. At other times, the test set is generated under conditions other than those of the training set; for example, in an image classification task, the test data may show different lighting conditions compared to the controlled laboratory conditions of the training data.

These datasets can suffer what is known as *dataset shift*, which is defined as "a challenging situation where the joint distribution of inputs and outputs differs between the training and test stages" [38]. This concept is broadly studied in the book by Quiñonero et al. [38]. Moreno et al. [39] state that there is no standard term to refer to this situation; therefore, there are numerous terms in the bibliography, such as "concept shift" or "concept drift," "changes of classification," "changing environments," etc. The authors suggest maintaining the term "dataset shift" for "any situation in which training and test data follow distributions that are in some way different." Then, different kinds of dataset shift may appear; for example, "covariate shift" is associated with those datasets where only the input distribution changes [38], while the conditional distribution of the outputs given the inputs remains unchanged. On the contrary, "concept shift" refers to different data distributions associated with changes in the class definitions.

Numerous alternatives to address this situation can be found in [38, 39]. However, it has to be noted that this problem may hinder the process of feature selection and classification.

1.3.5 Noisy Data

It is almost inevitable that there is some noise in most of the collected data, except in the most structured and synthetic environments. This "imperfect data" can be due to many sources, for instance, faulty measuring devices, transcription errors, and transmission irregularities. However, the accuracy of a classifier may greatly depend on the quality of the data used during the training phase, so a classifier built from a noisy training set might be less accurate and less compact than one built from the noise-free version of the same dataset using an identical algorithm [40].

Imperfections in a dataset can be dealt with in four broad ways: 1) leave the noise in, 2) data cleaning, i.e., filter the noise out, 3) data transformation, i.e, correct the noise, and 4) data reduction, that is, reduce the amount of data by aggregating values or removing and clustering redundant attributes [41]. In the first approach, the dataset is taken as is, with the noisy instances left in place. Algorithms that make use of the data are designed to be robust; that is, they can tolerate a certain amount of noise in the data. One approach is to develop robust algorithms that allow for noise by avoiding overfitting the model to the data, for example, using pruning techniques in decision trees [42]. In the second approach, instances that are suspected of being noisy according to certain evaluation criteria are discarded. A classifier is then built using only the retained instances in a smaller but cleaner dataset. [43] explores four techniques intended for data cleaning to enhance data analysis in the presence of high noise levels. However, there are two weaknesses in using this approach. First, by eliminating the whole instance, potentially useful information could also be discarded, such as the uncorrupted feature values. Secondly, when there is a large amount of noise in the dataset, the amount of information in the remaining clean dataset may not be sufficient for building a good classifier. These drawbacks lead us to the third approach that attempts to correct the tuples, so the noisy instances are identified, but instead of tossing these instances out, they are repaired by replacing the corrupted values with more appropriate ones. These corrected instances are then reintroduced into the dataset.

Regarding the fourth approach, dimensionality reduction—and consequently feature selection—is one of the most popular techniques to remove noisy (i.e., irrelevant) and redundant features. However, while feature selection has been the target of much research, very little study has been done to systematically address the issue of feature relevance in the presence of noisy data.

1.3.6 Outliers

Hawkins defines an outlier as “an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism” [44]. Outlier detection algorithms are useful in areas such as machine learning, data mining, pattern recognition, data cleansing and data warehousing [45] and have been of interest for many real applications such as credit card fraud detection (where purchasing behavior of a credit card owner usually changes when the card is stolen) or clinical applications (because unusual symptoms or test results may indicate potential health problems of a patient), network intrusion, sport performance analysis to identify outstanding players having abnormal parameter values, and other data-mining tasks [46]. Outlier detection methods can be divided into univariate methods and multivariate methods. Another taxonomy of outlier detection methods is between parametric (statistical) methods and nonparametric methods that are model-free [46]. There is a vast literature on outlier detection including statistical works, as well as probabilistic and Bayesian techniques attempting to find the model of the anomalies, variations of support vector machines or distance-based methods [45]. In [46] a comparison between different methods can be found.

1.3.7 Feature Cost

Test costs and misclassification costs are the two most important types of cost in cost-sensitive learning. The test cost is money, time, or other resources we pay for collecting a data item of an object. The misclassification cost is the penalty we receive after deciding that an object belongs to an incorrect class [47]. Feature selection works have mainly considered misclassification costs. However, most feature selection methods proposed in the literature do not cope with the test cost, although acquiring features in real-world applications may vary notably from one feature to another. For example, in medical applications, the feature acquisition can demand a clinical test that, in addition to its economics cost, poses a risk to the patient, or demand demographic data obtained from giving him a short questionnaire. In image texture analysis, the feature extraction process represents a significant computational cost that also depends on the method used [48].

1.4 A Guide for the Reader

The complexity and volume of data available in the field of machine learning is growing daily. This means that the number of research studies in this area, and specifically in the field of feature selection, is increasing. Although the authors’ aspiration was to address the feature selection field in depth, the scope of the book has been set to a particular class of problems: supervised classification. However, there are other fields in which feature selection is also being successfully applied, such as regression problems, unsupervised classification, clustering, etc. The next paragraph outlines the structure of this book.

Chapter 2 presents the foundations of feature selection, as well as a description of state-of-the-art feature selection methods. Then, Chapter 3 reviews the most popular methods in the literature and checks their performance in an artificial, controlled scenario, proposing some guidelines about their appropriateness in different domains. Chapter 4 analyzes the up-to-date contributions of feature selection research applied to the field of DNA microarray classification, whereas Chapter 5 is devoted to proving the benefits of feature selection in other real applications, such as the detection of intrusions in networks and the classification of the tear film lipid layer. Chapter 6 states the emerging challenges that researchers in feature selection must embrace in the following years and, finally, Appendix A presents the materials and methods used throughout the experiments performed in this book.

References

1. Zhai, Y., Ong, Y. and Tsang, I. The emerging “Big Dimensionality”. *IEEE Computational Intelligence Magazine*, 9(3):14–26, 2014.
2. Tan, M., Tsang, I. W. and Wang, L. Towards ultrahigh dimensional feature selection for big data. *Journal of Machine Learning Research*, 15:1371–1429, 2014.
3. Attenberg, J., Dasgupta, A., Langford, J., Smola, A. and Weinberger, K. Feature hashing for large scale multitask learning. *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120, 2009.
4. Donoho, D. L. and others. High-dimensional data analysis: The curses and blessings of dimensionality. *AMS Math Challenges Lecture*, pages 1–32, 2000.
5. Bellman, R. E. *Adaptive Control Processes: A Guided Tour*, volume 4. Princeton University Press, 1961.
6. Guyon, I. and Elisseeff, A. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.
7. Boutsidis, C., Drineas, P. and Mahoney, M. W. Unsupervised feature selection for the *k*-means clustering problem. In *Advances in Neural Information Processing Systems*, pages 153–161, 2009.
8. Roth, V. and Lange, T. Feature selection in clustering problems. In *Advances in Neural Information Processing Systems*, 2003.
9. Leardi, R. and Lupiáñez González, A. Genetic algorithms applied to feature selection in PLS regression: how and when to use them. *Chemometrics and Intelligent Laboratory Systems*, 41(2):195–207, 1998.
10. Paul, D., Bair, E., Hastie, T. and Tibshirani, R. ”Preconditioning” for feature selection and regression in high-dimensional problems. *The Annals of Statistics*, pages 1595–1618, 2008.
11. Dash, M. and Liu, H. Feature selection for classification. *Intelligent Data Analysis*, 1(3):131–156, 1997.
12. Pal, M. and Foody, G. M. Feature selection for classification of hyperspectral data by SVM. *IEEE Transactions on Geoscience and Remote Sensing*, 48(5):2297–2307, 2010.
13. Bonev, B. *Feature Selection Based on Information Theory*. Ph.D. thesis, University of Alicante, June 2010.
14. Hughes, G. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1):55–63, 1968.

15. Miller, A. J. Selection of subsets of regression variables. *Journal of the Royal Statistical Society. Series A (General)*, pages 389–425, 1984.
16. Guyon, I., Gunn, S., Nikravesh, M. and Zadeh, L. *Feature Extraction: Foundations and Applications*. Springer, 2006.
17. Liu, H. and Motoda, H. *Computational Methods of Feature Selection*. CRC Press, 2007.
18. Zhao, Z. A. and Liu, H. *Spectral Feature Selection for Data Mining*. Chapman & Hall/CRC, 2011.
19. Fan, J. and Li, R. Statistical challenges with high dimensionality: Feature selection in knowledge discovery. *arXiv preprint math/0602133*, 2006.
20. Fukunaga, K. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.
21. Tibshirani, R., Hastie, T., Narasimhan, B. and Chu, G. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences*, 99(10):6567–6572, 2002.
22. Fan, J. and Fan, Y. High dimensional classification using features annealed independence rules. *Annals of Statistics*, 36(6):2605, 2008.
23. Peng, H., Long, F. and Ding, C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005.
24. Hua, J. Tembe, W. D. and Dougherty, E. R. Performance of feature-selection methods in the classification of high-dimension data. *Pattern Recognition*, 42(3):409–424, 2009.
25. Ling, C. X. and Sheng, V. S. Class Imbalance Problem. *Encyclopedia of Machine Learning*, 171–171, 2010.
26. Galar, M., Fernandez, A., Barrenechea, E., Bustince, H. and Herrera, F. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(4):463–484, 2012.
27. Weiss, G. M. Mining with rarity: a unifying framework. *ACM SIGKDD Explorations Newsletter*, 6(1):7–19, 2004.
28. He, H. and Garcia, E. A. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.
29. Sun, Y., Wong, Andrew K. C. and Kamel, M. S. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(04):687–719, 2009.
30. López, V., Fernández, A., García, S., Palade, V. and Herrera, F. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250(0):113 – 141, 2013.
31. Chawla, N. V., Bowyer, K. W., Hall, L. O. and Kegelmeyer, W. P. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.
32. Cieslak, D. A. and Chawla, N. V. Learning decision trees for unbalanced data. In *Machine Learning and Knowledge Discovery in Databases*, pages 241–256. Springer, 2008.
33. Kailath, T. The divergence and Bhattacharyya distance measures in signal selection. *IEEE Transactions on Communication Technology*, 15(1):52–60, 1967.
34. López, V., Fernández, A., Moreno-Torres, J. G. and Herrera, F. Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics. *Expert Systems with Applications*, 39(7):6585–6608, 2012.
35. Basu, M and Ho, T. K. *Data Complexity in Pattern Recognition*. Springer, 2006.
36. Ho, T. K. and Basu, M. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):289–300, 2002.
37. KDD Cup 99 Dataset. `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html` [Last access: November 2014].
38. Quionero-Candela, J., Sugiyama, M., Schwaighofer, A. and Lawrence, N.D. *Dataset Shift in Machine Learning*. The MIT Press, 2009.
39. Moreno-Torres, J. G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N. V. and Herrera, F. A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530, 2012.

40. Teng, C.M. Combining noise correction with feature selection. *Data Warehousing and Knowledge Discovery, Lecture Notes in Computer Science*, 2737:340–349, 2003.
41. Tan, K. C., Teoh, E. J., Yu, Q. and Goh, K. C. A hybrid evolutionary algorithm for attribute selection in data mining. *Expert Systems with Applications*, 36(4):8616–8630, 2009.
42. Quinlan, J. R. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3):221–234, 1987.
43. Xiong, H., Pandey, G., Steinbach, M. and Kumar, V. Enhancing data analysis with noise removal. *IEEE Transactions on Knowledge and Data Engineering*, 18(3):304–319, 2006.
44. Hawkins, D. M. Identification of Outliers. *Monographs on Applied Probability and Statistics*, volume 11. Springer, 1980.
45. Escalante, H. J. A comparison of outlier detection algorithms for machine learning. *Proceedings of the International Conference on Communications in Computing*, 228–237, 2005.
46. Ben-Gal, I. Outlier detection. In *Data Mining and Knowledge Discovery Handbook. Second Edition*, pages 117–130. Springer, 2010.
47. Zhao, H., Min, F. and Zhu, W. Cost-sensitive feature selection of numeric data with measurement errors. *Journal of Applied Mathematics*, 2013.
48. Bolón-Canedo, V., Porto-Díaz, I., Sánchez-Maroño, N. and Alonso-Betanzos, A. A framework for cost-based feature selection. *Pattern Recognition*, 47(7), pages 2481–2489, 2014.

Chapter 2
Foundations of Feature Selection

Abstract In order to confront the problem of the high dimensionality of data, feature selection algorithms have become indispensable components of the learning process. Therefore, a correct selection of the features can lead to an improvement of the inductive learner in terms of learning speed, generalization capacity or simplicity of the induced model. A global overview of the feature selection process is given in Section 2.1. Then, Section 2.2 describes the different types of feature selection methods, as well as providing a description of the most popular algorithms for further analysis and explanation in subsequent chapters of the book. Finally, Section 2.3 summarizes this chapter.

In the few last years, several datasets with high dimensionality have become publicly available on the Internet. This fact has posed an interesting challenge to the research community, since for the machine learning methods it is difficult to deal with a large number of input features. At present, the dimensionality of the benchmark datasets that can be found in several data repositories has increased to millions, or even more. In fact, in [1], one of the analyses carried out by the authors reveals that seven out of 11 datasets that have appeared since 2007 exhibit dimensionality in the millions of features. To confront the problem of the large number of input features, dimensionality reduction techniques are indispensable and may help to improve learning performance. The term "Big Dimensionality" has been coined to refer to this issue, in contrast to the "Big Data" problem that deals with extremely huge sample size [1].

These dimensionality reduction techniques usually come in two flavors, *feature selection* and *feature extraction*, and each of them has its own merits [2]. On the one hand, feature extraction techniques achieve dimensionality reduction by combining the original features. In this manner, they are able to generate a set of new features, which is usually more compact and of stronger discriminating power. It is preferable in applications such as image analysis, signal processing, and information retrieval, where model accuracy is more important than model interpretability. On the other hand, feature selection achieves dimensionality reduction by removing the irrelevant and redundant features [3]. It is widely used in data mining applications, such as text mining, genetics analysis, and sensor data processing. Due to the fact that feature selection maintains the original features, it is especially useful for applications where the original features are important for model interpreting and knowledge extraction. This chapter will present the foundations of feature selection.

2.1 Feature Selection

Feature selection can be defined as the process of detecting the relevant features and discarding the irrelevant and redundant ones with the goal of obtaining a subset of features that describes properly the given problem with a minimum degradation of performance. It has several advantages [3], such as:

- Improving the performance of the machine learning algorithms.
- Data understanding, gaining knowledge about the process and perhaps helping to visualize it.
- General data reduction, limiting storage requirements and perhaps helping in reducing costs.
- Feature set reduction, saving resources in the next round of data collection or during utilization.
- Simplicity, possibility of using simpler models and gaining speed.

To all this, we will add that in the present scenario of Big Data analytics, feature selection plays a central role.

2.1.1 Feature Relevance

Intuitively, it can be determined that a feature is relevant if it contains some information about the target. More formally, Kohavi and John classified features into three disjoint categories, namely, strongly relevant, weakly relevant, and irrelevant features [4]. In their approach, the relevance of a feature X is defined in terms of an ideal Bayes classifier. A feature X is considered to be *strongly relevant* when the removal of X results in a deterioration of the prediction accuracy of the ideal Bayes classifier. A feature X is said to be *weakly relevant* if it is not strongly relevant and there exists a subset of features S, such that the performance of the ideal Bayes classifier on S is worse than the performance on $S \cup \{X\}$. A feature is defined as *irrelevant* if it is neither strongly nor weakly relevant.

2.1.2 Feature Redundancy

A feature is usually considered as redundant in terms of feature correlation [5]. It is widely accepted that two features are redundant to each other if their values are completely correlated, but it might not be so easy to determine feature redundancy when a feature is correlated with a set of features. According to [5], a feature is redundant and hence should be removed if it is weakly relevant and has a Markov blanket [6] within the current set of features. Since irrelevant features should be removed anyway, they are excluded from this definition of redundant features.

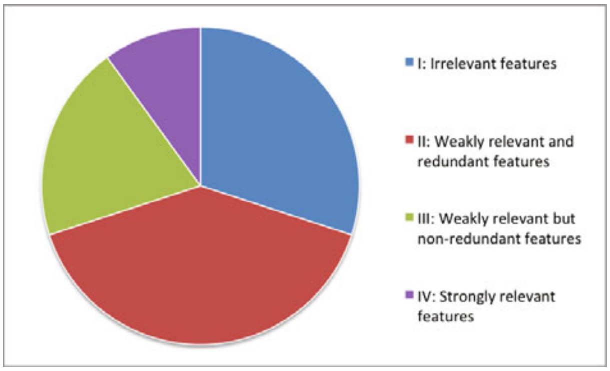

I: Irrelevant features
II: Weakly relevant and redundant features
III: Weakly relevant but non-redundant features
IV: Strongly relevant features

Fig. 2.1: Overview of feature relevance and redundancy

Figure 2.1 displays an overview of the relationship between feature relevance and redundancy. The entire feature set can be conceptually divided into four basic disjoint parts: irrelevant features (I), weakly relevant and redundant features (II), weakly relevant but nonredundant features (III) and strongly relevant features (IV) [5]. Notice that the optimal subset would contain all the features in parts III and IV.

2.2 Feature Selection Methods

Feature selection methods can be divided according to two approaches: *individual evaluation* and *subset evaluation* [5]. Individual evaluation is also known as feature ranking and assesses individual features by assigning them weights according to their degrees of relevance. On the other hand, subset evaluation produces candidate feature subsets based on a certain search strategy. Each candidate subset is evaluated by a certain evaluation measure and compared with the previous best one with respect to this measure. While the individual evaluation is incapable of removing redundant features because redundant features are likely to have similar rankings, the subset evaluation approach can handle feature redundancy with feature relevance.

However, methods in this framework can suffer from an inevitable problem caused by searching through all feature subsets required in the subset generation step, and thus, both approaches are worth studying.

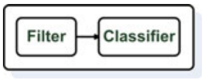
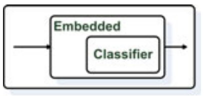
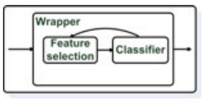
Aside from this classification, three major approaches can be distinguished based upon the relationship between a feature selection algorithm and the inductive learning method used to infer a model [3]:

- *Filters*, which rely on the general characteristics of training data and carry out the feature selection process as a preprocessing step with independence of the induction algorithm. This model is advantageous for its low computational cost and good generalization ability.
- *Wrappers*, which involve a learning algorithm as a black box and consist of using its prediction performance to assess the relative usefulness of subsets of variables. In other words, the feature selection algorithm uses the learning method as a subroutine with the computational burden that comes from calling the learning algorithm to evaluate each subset of features. However, this interaction with the classifier tends to give better performance results than filters.
- *Embedded methods*, which perform feature selection in the process of training and are usually specific to given learning machines. Therefore, the search for an optimal subset of features is built into the classifier construction and can be seen as a search in the combined space of feature subsets and hypotheses. This approach is able to capture dependencies at a lower computational cost than wrappers.

Table 2.1 provides a summary of the characteristics of the three feature selection methods, indicating the most prominent advantages and disadvantages.

Considering that several algorithms exist for each one of the previously commented approaches, there is a vast body of feature selection methods. Most researchers agree that "the best method" simply does not exist and their efforts are focused on finding a good method for a specific problem setting. In that sense, different methods have been developed to deal with large-scale datasets where the importance of feature selection is indisputable, since it is essential to minimize training time and allocated memory while maintaining accuracy. Nevertheless, it is important to bear in mind that most feature selection methods use the performance of the learned model as part of the selection process. In fact, from the three categories shown above (filters, wrappers and embedded), only filters are algorithm-independent. This property makes filters computationally simple and fast, and able to handle extremely large-scale datasets. However, most filters are univariate, i.e., they consider each feature independently of other features, a drawback that can be overcome by multivariate techniques which usually demand more computational resources.

Table 2.1: Feature selection techniques

Method	Advantages	Disadvantages
Filter 	Independence of the classifier Lower computational cost than wrappers Fast Good generalization ability	No interaction with the classifier
Embedded 	Interaction with the classifier Lower computational cost than wrappers Captures feature dependencies	Classifier-dependent selection
Wrapper 	Interaction with the classifier Captures feature dependencies	Computationally expensive Risk of overfitting Classifier-dependent selection

2.2.1 *Filter Methods*

Filter methods are based on performance evaluation metrics calculated directly from the data, without direct feedback from predictors that will finally be used on data with a reduced number of features [3]. As mentioned above, these algorithms are usually computationally less expensive than wrappers or embedded methods. In this subsection, the most popular filters are described, which will be used throughout this book.

2.2.1.1 Chi-Squared

This is an univariate filter based on the χ^2 statistic [7] which evaluates each feature independently with respect to the classes. The higher the value of chi-squared, the more relevant is the feature with respect to the class. Given a number of intervals (V), the number of classes (B), and the total number of instances (N), the value of chi-squared for a feature is calculated as

$$\chi^2 = \sum_{i=1}^{V} \sum_{j=1}^{B} \frac{[A_{ij} - \frac{R_i * B_j}{N}]^2}{\frac{R_i * B_j}{N}}, \tag{2.1}$$

where R_i denotes the number of instances in the range ith, B_j the number of instances in class jth, and A_{ij} the number of instances in the range ith and class jth.

2.2.1.2 Information Gain

The Information Gain filter [8] is one of the most common univariate methods of evaluation attributes. This filter evaluates the features according to their information gain and considers a single feature at a time. The entropy measure is considered as a measure to rank variables. The entropy of a class feature Y is

$$H(Y) = -\sum p(y)log_2(p(y)), \tag{2.2}$$

where $p(y)$ is the marginal probability density function for the random variable Y. If the observed values of Y in the training dataset S are partitioned according to the values of a second feature X, and the entropy of Y with respect to the partitions induced by X is less than the entropy of Y prior to partitioning, then there is a relationship between features Y and X. Then the entropy of Y after observing X is

$$H(Y|X) = \sum p(x) \sum p(y|x)log_2(p(y|x)), \tag{2.3}$$

where p(y|x) is the conditional probability of y given x. Given the entropy as a criterion of "impurity" in a training set S, we can define a measure reflecting additional information about Y provided by X that represents the amount by which the entropy of Y decreases. This measure, an indicator of the dependency between X and Y, is known as IG:

$$IG = H(Y) - H(Y|X) = H(X) - H(X|Y). \tag{2.4}$$

IG is a symmetrical measure. The method provides an orderly classification of all the features, and then a threshold is required to select a certain number of them according to the order obtained. A weakness of the IG criterion is that it is biased in favor of features with more values even when they are not more informative.

2.2.1.3 Correlation-Based Feature Selection, CFS

This is a simple multivariate filter algorithm that ranks feature subsets according to a correlation-based heuristic evaluation function [9]. The bias of the evaluation function is toward subsets that contain features that are highly correlated with the class and uncorrelated with each other. Irrelevant features should be ignored because they will have low correlation with the class. The measure used is

$$Q_{zc} = (m\overline{Q_{zi}})/\sqrt{m + m(m-1)\overline{Q_{ii}}}, \tag{2.5}$$

where Q_{zc} is the correlation between the individual features and the output class, m is the number of features, Q_{zi} is the measure of correlation between the individual features and the output variable, and Q_{ii} is the average intercorrelation among features. So, the measure used assigns high values to the subsets that are highly correlated with the output while being weakly correlated with each other. Irrelevant features should be ignored because they will have low correlation with the class. Redundant features should be screened out as they will be highly correlated with one or more of the remaining features. The acceptance of a feature will depend on the extent to which it predicts classes in areas of the instance space not already predicted by other features. The numerator in equation (2.5) is an indicator of the prediction capabilities of a feature subset, while the denominator can be seen as an index of the redundancy among them in the subset. As each feature is treated individually, CFS is not capable of identifying strong interactions, as those that can appear in parity problems. However, the algorithm achieves good results in those problems in which interaction is not high. Applying the algorithm requires two steps. On the first step, the continuous features are discretized, and subsequently a measure called Symmetrical Uncertainty (SU) developed by [10] is used. SU is defined as

$$SU(X;Y) = IG(X;Y)/(H(X)+H(Y)). \tag{2.6}$$

Generally, SU is normalized between 0 and 1, and it is used as a measure of correlation between features and the concept target; then features are weighted by their corresponding SU value; the larger the SU value, the higher the weight.

2.2.1.4 Consistency-Based Filter

The filter based on consistency [11] evaluates the worth of a subset of features by the level of consistency in the class values when the training instances are projected onto the subset of attributes. From the space of features, the algorithm generates a random subset S in each iteration. If S contains fewer features than the current best subset, the inconsistency index of the data described by S is compared with the index of inconsistency in the best subset. If S is as consistent as or more than the best subset, S becomes the best subset. The criterion of inconsistency, which is the key to the success of this algorithm, specifies how large the reduction of dimension in the data can be. If the rate of consistency of the data described by selected characteristics is less than a set threshold, it means that the reduction in size is acceptable. Notice that this method is multivariate.

2.2.1.5 Fast Correlation-Based Filter, FCBF

The fast correlated-based filter method [12] is a multivariate algorithm that measures feature-class and feature-feature correlation. FCBF starts by selecting a set of features that is highly correlated with the class based on symmetrical uncertainty (SU) (see equation 2.6), which is defined as the ratio between the information gain and the entropy of two features. Then, it applies three heuristics that remove the redundant features and keep the features that are more relevant to the class. FCBF was designed for high-dimensionality data and has been shown to be effective in removing both irrelevant and redundant features. However, it fails to take into consideration the interaction between features.

2.2.1.6 INTERACT

The INTERACT algorithm [13] uses the same goodness measure as the FCBF filter, i.e., SU (equation 2.6), but it also includes the consistency contribution, which is an indicator of how significantly the elimination of a feature will affect consistency. The algorithm consists of two major parts. In the first part, the features are ranked in descending order based on their SU values. In the second part, features are evaluated one by one starting from the end of the ranked feature list. If the consistency contribution of a feature is less than an established threshold, the feature is removed; otherwise it is selected. Their authors stated that this method can handle feature interaction, and efficiently selects relevant features.

2.2.1.7 ReliefF

The filter ReliefF [14] is an extension of the original Relief algorithm [15], which can handle multiclass problems and is more robust and capable of dealing with incomplete and noisy data. As the original Relief, ReliefF works by randomly selecting an instance R_i from the data and then locating its k nearest neighbors from the same (*nearest hits*, H_j) class and k nearest neighbors from each one of the other different classes, *nearest misses* $M_j(C)$. It updates the quality estimation $W[A]$ for all attributes A depending on their values for R_i, hits H_j and misses $M_j(C)$. If instances R_i and H_j have different values of the attribute A, then this attribute separates instances of the same class, which clearly is not desirable, and thus the quality estimation $W[A]$ has to be decreased. On the contrary, if instances R_i and M_j have different values of the attribute A for a class then the attribute A separates two instances with different class values—which is desirable—so the quality estimation $W[A]$ is increased. Since ReliefF considers multiclass problems, the contribution of all the hits and all the misses is averaged. Besides, the contribution for each class of the misses is weighted with the prior probability of that class $P(C)$ (estimated from the training set). The whole process is repeated t times, where t is a user-defined parameter (see Algorithm 2.1).

The function $diff(A, I_1, I_2)$ calculates the difference between the values of the attribute A for two instances, I_1 and I_2. This method may be applied in all situations, has low bias, includes interaction among features and may capture local dependencies which other methods miss.

2.2.1.8 Minimum Redundancy Maximum Relevance, mRMR

The mRMR method [16] is a multivariate filter that selects features that have the highest relevance to the target class and are also minimally redundant, i.e., selects features that are maximally dissimilar to each other. Both optimization criteria (Maximum-Relevance and Minimum-Redundancy) are based on mutual information. As a multivariate filter, mRMR models feature dependencies and detects redun-

Algorithm 2.1: Pseudocode of ReliefF algorithm

```
[htp] Data: training set $D$, iterations $t$, attributes $a$
Result: the vector $W$ of estimations of the qualities of attributes
1 set all weights $W[A] := 0$
2 for $i \leftarrow 1$ to $m$ do
3     randomly select an instance $R_i$
4     find $k$ nearest hits $H_j$
5     for each class $C \neq class(R_i)$ do
6         from class $C$ find $k$ nearest misses $M_j(C)$
      end
  end
7 for $f \leftarrow 1$ to $a$ do
8     $W[f] := W[f] - \frac{\sum_{j=1}^{k} diff(f,R_i,H_j)}{(t\cdot k)} + \frac{\sum_{C\neq class(R_i)}\left[\frac{P(C)}{1-P(class(R_i))}\sum_{j=1}^{k} diff(f,R_i,M_j(C))\right]}{(t\cdot k)}$
  end
```

dancy, but at the cost of being slower and less scalable than univariate techniques. As stated above, the mRMR method can rank features based on their relevance to the target, and at the same time, the redundancy of features is also considered. Features that have the best trade-off between maximum relevance to target and minimum redundancy are considered good features. In terms of mutual information, the purpose of feature selection is to find a feature set S with m features $\{x_i\}$, which jointly have the largest dependency on the target class Y. This scheme, called *maximum dependency*, has the form

$$max\, D(S,Y),\, D = I(\{x_i, i = 1,...,m\};Y).$$

Obviously, when m equals 1, the solution is the feature that maximizes $I(x_j;Y)(1 \leq j \leq M)$. When $m > 1$, a simple incremental search scheme is to add one feature at a time: given the set with $m-1$ features, $S_m - 1$, the m^{th} feature can be determined as the one that contributes to the largest increase of $I(S;Y)$, which takes the form

$$I(S_m;Y) = \iint p(S_m;Y)log\frac{p(S_m,Y)}{p(S_m)p(Y)}dS_m dY.$$

Despite the theoretical value of *maximum dependency*, it is often hard to get an accurate estimation for multivariate density $p(x_1,...,x_m)$ and $p(x_1,...,x_m;Y)$, because of two difficulties in the high-dimensional space: 1) the number of samples is often insufficient and 2) the multivariate density estimation often involves computing the inverse of the high-dimensional covariance matrix, which is usually an ill-posed problem. Another drawback of *maximum dependency* is the slow computational speed. These problems are most pronounced for continuous feature variables. Even for discrete (categorical) features, the practical problems in implementing *maximum dependency* cannot be completely avoided. For example, suppose each feature has three categorical states and N samples. K features could have a maximum of $min(3^K;N)$ joint states. When the number of joint states increases very quickly and becomes comparable to the number of samples, N, the joint probability of these features, as well as the mutual information, cannot be estimated correctly. Hence, although *maximum dependency* feature selection might be useful for selecting a very small number of features when N is large, it is not appropriate for applications where the aim is to achieve high classification accuracy with a reasonably compact set of features. As a *maximum dependency* criterion is hard to implement, an alternative is to select features based on *maximal relevance* criterion (*maximum relevance*). Maximum relevance is to search for features satisfying equation (2.7), which approximates the *max-dependency* with the mean value of all mutual information values between individual feature x_i (of a feature set S) and class Y

$$maxD(S,Y), D = \frac{1}{|S|} \sum_{x_i \in S} I(x_i;Y). \tag{2.7}$$

It is likely that features selected according to maximum relevance could have rich redundancy, i.e., the dependency among these features could be large. When two features highly depend on each other, the respective class-discriminative power would not change much if one of them were removed. Therefore, the following minimal redundancy (Min-Redundancy) condition can be added to select mutually exclusive features:

$$min\ R(S),\ R = \tfrac{1}{|S|^2} \textstyle\sum_{x_i,x_j \in S} I(x_i,x_j).$$

The criterion combining the above two constraints is called "minimal-redundancy-maximal-relevance" (mRMR). The operator $\Phi(D,R)$ is defined to combine D and R and considers the following simplest form to optimize D and R simultaneously:

$$max\ \Phi(D,R),\ \Phi = D - R.$$

In practice, incremental search methods can be used to find the near-optimal features defined by $\Phi(.)$ Suppose we already have S_{m-1}, the feature set with $m-1$ features. The task is to select the m^{th} feature from the set $\{X - S_{m-1}\}$. This is done by selecting the feature that maximizes $\Phi(.)$. The respective incremental algorithm optimizes the following condition:

$$max_{x_j \in X - S_{m-1}} [I(x_j;Y) - \tfrac{1}{m-1} \textstyle\sum_{x_i \in S_{m-1}} I(x_j;x_i)].$$

2.2.1.9 $\mathscr{M}_d$

The $\mathscr{M}_d$ filter [17] is an extension of mRMR that uses a measure of monotone dependence (instead of mutual information) to assess relevance and irrelevance, since it is simpler to estimate from data (compared to Mutual Information, MI), and it carries many desired properties of a measure of dependence, that is, it is bounded, symmetric, and reaches its maximum if two random variables share a monotonic relationship. One of its contributions is the inclusion of a free parameter (λ) that controls the relative emphasis given on relevance and redundancy. In Chapter 3, two values of lambda will be tested: 0 and 1. When λ is equal to 0, the redundancy effect disappears and the measure is based only on maximizing the relevance. On the other hand, when λ is equal to 1, it is more important to minimize the redundancy among variables. These two values of λ were chosen because we are interested in checking the performance of the method when the effect of the redundancy disappears. Also, Seth and Principe [17] stated that $\lambda = 1$ performs better than other λ values.

Table 2.2 reports the main characteristics of the filters employed in this book. With regard to the computational cost, it can be noticed that some of the proposed filter techniques are univariate. This means that each feature is considered separately, thereby ignoring feature dependencies, which may lead to worse classification performance when compared to other types of feature selection techniques. However, they have the advantage, in theory, of being scalable. Multivariate filter techniques were introduced, aiming to incorporate feature dependencies to some degree, but at the cost of reducing their scalability.

Table 2.2: Summary of filters

	Uni/Multivariate	Ranker/Subset
Chi-Squared	Univariate	Ranker
Information Gain	Univariate	Ranker
ReliefF	Multivariate	Ranker
mRMR	Multivariate	Ranker
$\mathscr{M}_d$	Multivariate	Ranker
CFS	Multivariate	Subset
FCBF	Multivariate	Subset
INTERACT	Multivariate	Subset
Consistency	Multivariate	Subset

2.2.1.10 A Framework for Information Theoretic Feature Selection

Recently, Brown et al. [18] have presented a unifying framework for information theoretic feature selection, bringing almost two decades of research on heuristic scoring criteria under a single theoretical interpretation. They proposed a generic formula to express several selection criteria based on information theory (such as IG or mRMR) by adjusting its parameters:

$$J = I(X_i;Y) - \beta \sum_{j \in S} I(X_j;X_i) + \gamma \sum_{j \in S} I(X_j;X_i|Y). \tag{2.8}$$

In this formula, one can distinguish the following elements:

Relevance It is computed as $I(X_i;Y)$; i.e., mutual information between a given feature X_i and the class label Y.

Redundancy It is computed as $\sum_{j\in S} I(X_j;X_i)$; i.e., mutual information between a given feature X_i and the already selected features $X_j \in S$.

Conditional redundancy It is computed as $\sum_{j\in S} I(X_j;X_i|Y)$; i.e., mutual information between a given feature X_i and the already selected features $X_j \in S$ conditioned by the class label Y.

Some of the criteria which can be expressed by modifying Eq. (2.8) can be seen in Table 2.3.

Table 2.3: Different criteria for feature selection based on mutual information and how to express them from the generic equation (2.8)

Name of criterium	
Original equation	Adaptation
Mutual Information Maximization (MIM) [19]	
$J_{mim}(X_i) = I(X_i;Y)$	$J_{mim} = I(X_i;Y) - 0\sum_{j\in S} I(X_j;X_i) + 0\sum_{j\in S} I(X_i;X_j\|Y)$
Mutual Information Feature Selection (MIFS) [20]	
$J_{mifs}(X_i) = I(X_i;Y) - \beta\sum_{X_j\in S} I(X_i;X_j)$	$J_{mifs} = I(X_i;Y) - \beta\sum_{j\in S} I(X_j;X_i) + 0\sum_{j\in S} I(X_i;X_j\|Y)$
Joint Mutual Information (JMI) [21]	
$J_{jmi}(X_i) = \sum_{X_j\in S} I(X_iX_j;Y)$	$J_{jmi} = I(X_i;Y) - \frac{1}{\|S\|}\sum_{j\in S} I(X_j;X_i) + \frac{1}{\|S\|}\sum_{j\in S} I(X_i;X_j\|Y)$
Conditional Mutual Information (CMI)	
$J_{cmi} = I(X_i;Y\|S)$	$J'_{cmi} = I(X_i;Y) - \sum_{j\in S} I(X_j;X_i) + \sum_{j\in S} I(X_i;X_j\|Y)$
minimum-Redundancy Maximum-Relevance (mRMR) [16]	
$J_{mrmr} = I(X_i;Y) - \frac{1}{\|S\|}\sum_{j\in S} I(X_j;X_i)$	$J_{mrmr} = I(X_i;Y) - \frac{1}{\|S\|}\sum_{j\in S} I(X_j;X_i) + 0\sum_{j\in S} I(X_i;X_j\|Y)$
Conditional Mutual Information Maximization (CMIM) [22]	
$J_{cmim} = \min_{X_j\in S}[I(X_i;Y\|X_j)]$	$J_{cmim} = I(X_i;Y) - \max_{j\in S}[I(X_j;X_i) - I(X_i;X_j\|Y)]$
Informative Fragments (IF) [23] (equivalent to CMIM)	
$J_{if} = \min_{X_j\in S}[I(X_iX_j;Y) - I(X_j;Y)]$	$J_{if} = J_{cmim} = I(X_i;Y) - \max_{j\in S}[I(X_j;X_i) - I(X_i;X_j\|Y)]$
Interaction Capping (ICAP) [24]	
$J_{icap} = I(X_i;Y) - \sum_{X_j\in S}\max[0, I(X_i;X_j) - I(X_i;X_j\|Y)]$	$J_{icap} = I(X_i;Y) - \sum_{X_j\in S}\max[0, I(X_i;X_j) - I(X_i;X_j\|Y)]$

2.2.2 Embedded Methods

In contrast to filter and wrapper approaches, embedded methods do not separate the learning from the feature selection part. Embedded methods include algorithms, which optimize a regularized risk function with respect to two sets of parameters:

the parameters of the learning machine and the parameters indicating which features are selected. The embedded methods used are subsequently described.

2.2.2.1 Recursive Feature Elimination for Support Vector Machines, SVM-RFE

This embedded method was introduced by Guyon in [25]. It performs feature selection by iteratively training an SVM classifier with the current set of features and removing the least important feature indicated by the weights in the SVM solution.

2.2.2.2 Feature Selection-Perceptron, FS-P

FS-P [26] is an embedded method based on a perceptron. A perceptron is a type of artificial neural network that can be seen as the simplest kind of feedforward neural network: a linear classifier. The basic idea of this method consists in training a perceptron in the context of supervised learning. The interconnection weights are used as indicators of which features could be the most relevant and provide a ranking.

2.2.3 Wrapper Methods

The idea of the wrapper approach is to select a feature subset using a learning algorithm as part of the evaluation function. Instead of using subset sufficiency, entropy or another explicitly defined evaluation function, a kind of "black box" function is used to guide the search. The evaluation function for each candidate feature subset returns an estimate of the quality of the model that is induced by the learning algorithm. This can be rather time consuming, since for each candidate feature subset evaluated during the search, the target learning algorithm is usually applied several times (e.g., in the case of ten-fold cross-validation used to estimate model quality). In this book the wrapper described as follows will be used; which can be combined with any learning algorithm.

In this book, the *WrapperSubsetEval* will be used, as it is provided by Weka [27], and can be combined with any learning algorithm. This wrapper evaluates attribute sets by using a learning scheme. Cross validation is used to estimate the accuracy of the learning scheme for a set of attributes. The algorithm starts with the empty set of attributes and searches forward, adding attributes until performance does not improve further.

2.2.4 Other Approaches

Numerous papers and books exist that prove the benefits of the feature selection process [2, 3, 4]. However, to the best of the authors' knowledge, none of the existing methods has demonstrated its superiority, regardless of the type of scenario. For this reason, a typical approach is to find a method that works satisfactorily for a specific problem setting. New feature selection methods are constantly emerging using different strategies: a) combining several feature selection methods, which could be done by using algorithms from the same approach, such as two filters [28], or coordinating algorithms from two different approaches, usually filters and wrappers [29, 30]; b) combining feature selection approaches with other techniques, such as feature extraction [31] or tree ensembles [32]; c) reinterpreting existing algorithms [33], sometimes to adapt them to specific problems [34]; d) creating new methods to deal with still unresolved situations [35, 36]; and e) using an ensemble of feature selection techniques to ensure better behavior [37, 38].

2.3 Summary

To confront the problem of the high dimensionality of data, feature selection algorithms have become indispensable components of the learning process. Hence, a correct selection of the features can lead to an improvement of the inductive learner in terms of learning speed, generalization capacity or simplicity of the induced model.

Feature selection methods may roughly be divided into three types: filters, wrappers and embedded methods. This chapter has summarized the advantages and disadvantages of each approach, as well as described the most popular algorithms that will be used throughout this book.

References

1. Zhai, Y., Ong, Y. and Tsang, I. The emerging "Big Dimensionality". *IEEE Computational Intelligence Magazine*, 9(3):14–26, 2014.
2. Zhao, Z. A. and Liu, H. *Spectral Feature Selection for Data Mining*. Chapman & Hall/CRC, 2011.
3. Guyon, I., Gunn, S., Nikravesh, M. and Zadeh, L. *Feature Extraction: Foundations and Applications*. Springer, 2006.
4. Kohavi, R. and John, G. H. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1):273–324, 1997.
5. Yu, L. and Liu, H. Efficient feature selection via analysis of relevance and redundancy. *The Journal of Machine Learning Research*, 5:1205–1224, 2004.
6. Koller, D. and Sahami, M. Toward optimal feature selection. In *13th International Conference on Machine Learning*, pages 284–292, 1995.

7. Liu, H. and Setiono, R. Chi2: Feature selection and discretization of numeric attributes. In *IEEE 7th International Conference on Tools with Artificial Intelligence*, pages 388–391, 1995.
8. Quinlan, J. R. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
9. Hall, M. A. *Correlation-Based Feature Selection for Machine Learning*. Ph.D. thesis, The University of Waikato, 1999.
10. Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. *Numerical Recipes in C* Cambridge University Press, Cambridge, 1988.
11. Dash, M. and Liu, H. Consistency-based search in feature selection. *Artificial Intelligence*, 151(1):155–176, 2003.
12. Yu, L. and Liu, H. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *International Conference on Machine Learning*, volume 20, pages 856–863, 2003.
13. Zhao, Z. and Liu, H. Searching for interacting features. In *International Joint Conference on Artificial Intelligence*, volume 7, pages 1156–1161, 2007.
14. Kononenko, I. Estimating attributes: Analysis and extensions of Relief. In *Machine Learning: ECML-94*, pages 171–182. Springer, 1994.
15. Kira, K. and Rendell, L. A. A practical approach to feature selection. In *9th International Workshop on Machine Learning*, pages 249–256. Morgan Kaufmann, 1992.
16. Peng, H., Long, F. and Ding, C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005.
17. Seth, S. and Principe, J. C. Variable selection: A statistical dependence perspective. In *IEEE 9th International Conference on Machine Learning and Applications*, pages 931–936, 2010.
18. Brown, G., Pocock, A., Zhao, M. J. and Luján, M. Conditional likelihood maximisation: A unifying framework for information theoretic feature selection. *The Journal of Machine Learning Research*, 13(1): 27–66, 2012.
19. Lewis, D. D. Feature selection and feature extraction for text categorization. In *Workshop on Speech and Natural Language*, pages 212–217, 1992.
20. Battiti, R. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4):537–550, 1994.
21. Yang, H. H. and Moody, J. E. Data visualization and feature selection: New algorithms for nongaussian data. In *Advances in Neural Information Processing Systems*, pages 687–702, 1999.
22. Fleuret, F. Fast binary feature selection with conditional mutual information. *The Journal of Machine Learning Research*, 13:1531–1555, 2004.
23. Vidal-Naquet, M. and Ullman, S. Object Recognition with Informative Features and Linear Classification. In *International Conference on Computer Vision*, pages 281, 2003.
24. Jakulin, A. *Attribute Interactions in Machine Learning*. Ph.D. Thesis, University of Ljubljana, 2003.
25. Guyon, I., Weston, J., Barnhill, S. and Vapnik, V. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.
26. Mejía-Lavalle, M., Sucar, E. and Arroyo, G. Feature selection with a perceptron neural net. In *International Workshop on Feature Selection for Data Mining*, pages 131–135, 2006.
27. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I. H. The Weka data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.
28. Zhang, Y., Ding, C. and Li, T. Gene selection algorithm by combining ReliefF and mRMR. *BMC Genomics*, 9(Supl 2):S27, 2008.
29. Peng, Y., Wu, A. and Jiang, J. A novel feature selection approach for biomedical data classification. *Journal of Biomedical Informatics*, 43(1):15–23, 2010.
30. El Akadi, A., Amine, A., El Ouardighi, A. and Aboutajdine, D. A two-stage gene selection scheme utilizing mRMR filter and GA wrapper. *Knowledge and Information Systems*, 26(3):487–500, 2011.
31. Vainer, I., Kraus, S., Kaminka, G. A. and Slovin, H. Obtaining scalable and accurate classification in large-scale spatio-temporal domains. *Knowledge and Information Systems*, 29(3):527–564, 2011.

32. Tuv, E., Borisov, A., Runger, G. and Torkkola, K. Feature selection with ensembles, artificial variables, and redundancy elimination. *The Journal of Machine Learning Research*, 10:1341–1366, 2009.
33. Sun, Y. and Li, Y. Iterative Relief for feature weighting. In *23rd International Conference on Machine Learning*, pages 913–920, 2006.
34. Sun, Y., Todorovic, S. and Goodison, S. A feature selection algorithm capable of handling extremely large data dimensionality. In *SIAM International Conference on Data Mining*, pages 530–540, 2008.
35. Chidlovskii, B. and Lecerf, L. Scalable feature selection for multi-class problems. In *Machine Learning and Knowledge Discovery in Databases*, pages 227–240, 2008.
36. Loscalzo, S., Yu, L. and Ding, C. Consensus group stable feature selection. In *15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 567–576. ACM, 2009.
37. Saeys, Y., Abeel, T. and Van de Peer, Y. Robust feature selection using ensemble feature selection techniques. In *Machine Learning and Knowledge Discovery in Databases*, pages 313–325, 2008.
38. Bolón-Canedo, V., Sánchez-Maroño, N. and Alonso-Betanzos, A. An ensemble of filters and classifiers for microarray data classification. *Pattern Recognition*, 45(1):531–539, 2012.

Chapter 3
A Critical Review of Feature Selection Methods

Abstract Feature selection has been a fruitful field of research and it is undoubtedly important. However, a statement like "the best feature selection method" simply does not exist in general, making it difficult for users to select one method over another. For this reason, the objective of this chapter is to perform a critical review of state-of-the-art feature selection methods. The chapter starts with the description of the existing reviews (Section 3.1). Then, Section 3.2 depicts the methods and data involved in the experiments of this chapter and Section 3.3 shows the results obtained. In Section 3.4 we present several cases of study, aiming at deciding between methods that showed similar behaviors. Finally, Section 3.5 analyzes and discusses the findings of the experimental study and Section 3.6 summarizes this chapter.

In the past few years, feature selection algorithms have become indispensable components of the learning process to get rid of irrelevant and redundant features. As mentioned in the previous chapter, two major approaches exist in feature selection: *individual evaluation* and *subset evaluation*. Individual evaluation is also known as feature ranking [1] and assesses individual features by assigning them weights according to their degrees of relevance. On the other hand, subset evaluation produces candidate feature subsets based on a certain search strategy. Besides this classification, feature selection methods can also be divided into three models: *filters*, *wrappers* and *embedded* methods [2]. With such a vast body of feature selection methods, it is necessary to determine some criteria that enable users to adequately decide which algorithm to use—or not—in certain situations.

There are several situations that can hinder the process of feature selection and subsequent classification, such as those presented in Section 1.3, such as the presence of irrelevant and redundant features, noise in the data or interaction between attributes. In the presence of hundreds or thousands of features, such as DNA microarray analysis, researchers notice [3, 4] that commonly a large number of features

Part of the content of this chapter was previously published in *Knowledge and Information Systems* (`doi:10.1007/s10115-012-0487-8`)

V. Bolón-Canedo et al., *Feature Selection for High-Dimensional Data*, Artificial Intelligence: Foundations, Theory, and Algorithms, DOI 10.1007/978-3-319-21858-8_3

are not informative because they are either irrelevant or redundant with respect to the class concept. Moreover, when the number of features is high but the number of samples is small, machine learning gets particularly difficult, since the search space will be sparsely populated and the model will not be able to correctly distinguish between the relevant data and the noise [5].

This chapter reviews the most popular feature selection methods in the literature and checks their performance in a controlled artificial experimental scenario. In this manner, the ability of the algorithms is contrasted to select the relevant features and to discard the irrelevant ones without permitting noise or redundancy to obstruct the process. A scoring measure will be introduced to compute the degree of matching between the output given by the algorithm and the known optimal solution, as well as the classification accuracy.

3.1 Existing Reviews of Feature Selection Methods

Feature selection, since it is an important activity in data preprocessing, has been widely studied in the past few years by machine learning researchers. This technique has found success in many different real-world applications like DNA microarray analysis [6], intrusion detection [7, 8], text categorization [9, 10] or information retrieval [11], including image retrieval [12] or music information retrieval [13].

Bearing in mind the large number of feature selection methods available, it is easy to note that carrying out a comparative study is an arduous task. Another problem is testing the effectiveness of these feature selection methods when real datasets are employed, usually without knowing the relevant features. In these cases the performance of the feature selection methods clearly rely on the performance of the learning method used afterwards and it can vary notably from one method to another. Moreover, performance can be measured using many different metrics such as computer resources (memory and time), accuracy, ratio of features selected, etc. Besides, datasets may include a great number of challenges: multiple class output, noisy data, huge number of irrelevant features, redundant or repeated features, ratio number of samples/number of features very close to zero and so on (see Chapter 1). It should be noted that a comparative study tackling all these considerations might be considered unapproachable, and therefore most of the interesting comparative studies are focused on the problem to be solved. So, for example, [9] presented an empirical comparison of 12 feature selection methods evaluated on a benchmark of 229 text classification problem instances; another comparative study [14] is used for breast cancer detection in mammograms. Other works are devoted to a specific approach [15], in which an experimental study of eight typical mutual information-based filters on 33 datasets is presented; or to an evaluation of the capability of the survival ReliefF algorithm (sReliefF) and of a tuned sReliefF approach to properly select the causative pair of attributes [16]. Similarly, there are works examining different feature selection methods to obtain good performance results using a specific classifier (naive Bayes in [17], C4.5 in [18] or the theoretical review for support vector machines, SVMs, in [19]). Related to dataset challenges, there are several works trying to address the problem of high dimensionality, in both dimensions (samples and features) or in one of them, i.e., a large number of features versus a low number of samples; most of these studies also tackle multiple class problems [20, 21, 22, 23, 24]. The majority of current real datasets (microarray, text retrieval, etc.) also present noisy data; however, no specific feature selection comparative studies dealing with this complex problem were found in the literature, although some interesting works have been proposed; see, for example, [25, 26, 27]. Focusing on nonlinear methods, it is worth mentioning the study of Guyon et al. [28]. From a theoretical perspective, Liu et al. [29] presented a survey of feature selection methods, providing some guidelines in selecting feature selection algorithms, paving the way to build an integrated system for intelligent feature selection. More recently, Brown et al. [30] empirically evaluated nine popular feature selection methods based on information theory on UCI datasets and on the well-known benchmark NIPS Feature Selection Challenge. Along the same line, Vergara and Estévez [31] presented a review of the state of the art of information-theoretic feature selection methods.

More experimental work on feature selection algorithms for comparative purposes can be found [32, 33, 34, 35, 36], some of which were performed over artificially generated data, like the widely-used *Parity*, *Led* or *Monks* problems [37]. Several authors choose to use artificial data since the desired output is known; therefore a feature selection algorithm can be evaluated independently of the classifier used. Although the final goal of a feature selection method is to test its effectiveness over a real dataset, the first step should be on synthetic data. The reason for this is twofold [38]:

1. Controlled experiments can be developed by systematically varying chosen experimental conditions, like adding more irrelevant features or noise to the input. This facilitates drawing more useful conclusions and testing the strengths and weaknesses of the existing algorithms.
2. The main advantage of artificial scenarios is the knowledge of the set of optimal features that must be selected; thus the degree of closeness to any of these solutions can be assessed in a confident way.

3.2 Experimental Settings

As was stated in Section 3.1, the first step to test the effectiveness of a feature selection method should be on synthetic data, since the knowledge of the optimal features and the chance to modify the experimental conditions allows more useful conclusions to be drawn. The datasets chosen for this study try to cover different problems: increasing the number of irrelevant features, redundancy, noise in the output, alteration of the inputs, nonlinearity of the data, etc. These factors complicate the task of the feature selection methods which, as will be shown afterwards, are very affected by them. Besides, some of the datasets have a significantly higher number of features than samples, which implies an added difficulty for the correct selection of the relevant features. The characteristics of the 11 synthetic datasets employed (Corral, Corral-100, XOR-100, Parity3+3, Led-25, Led-100, Monk3, SD1, SD2, SD3 and Madelon) can be found in Appendix A, Table A.1.

Twelve different feature selection methods are tested and compared in order for useful conclusions to be drawn. The chosen methods are CFS, consistency-based, INTERACT, Information Gain, ReliefF, mRMR, M_d, SVM-RFE, FS-P and the wrapper combined with SVM and C4.5; a description of all of them can be found in Chapter 2. In the case of M_d, two different values of λ will be tested: 0 and 1. As for SVM-RFE, two kernels will be considered: a linear kernel, which is the kernel used by default, and a Gaussian kernel in an attempt to solve more complex problems [39], which will be referred to in this chapter as SVM-RFE-G.

As previously mentioned in Chapter 2, two major approaches exist in feature selection: *individual evaluation*, which provides an ordered ranking of the features; and *subset evaluation*, which produces a candidate feature subset. When a ranking of the features is returned, it is necessary to establish a threshold in order to discard those less relevant to the algorithm. Unfortunately, where to establish the threshold is not an easy-to-solve question. Belanche et al. [38] opted for discarding those weights associated with the ranking which were further than two variances from the mean. On the other hand, Mejia et al. [40] use a threshold defined by the largest gap between two consecutive ranked attributes, and other authors [41] only studied the whole ranking paying more attention to the first ranked features. However, in this experimental study it is impossible to use a threshold related to the weights associated with the ranking, since some of the ranker methods (SVM-RFE, mRMR and M_d) eliminate chunks of features at a time and do not provide weights. To solve this problem and for the sake of fairness, in these experiments we heuristically set the following rules to decide the number of features that ranker methods should return, according to the number of total features (N):

- if $N < 10$, select 75% of the features
- if $10 < N < 75$, select 40% of the features
- if $75 < N < 100$, select 10% of the features
- if $N > 100$, select 3% of the features

At this point it should be clarified that the datasets SD (see Appendix A), due to their particularities, will be analyzed in a different manner which will be explained later. According to the rules shown above, the number of features that will be returned by ranker methods is five for the datasets Corral, Parity3+3 and Monk3; ten for the datasets Corral-100, XOR-100 and both versions of Led; and 15 for Madelon.

A scoring measure was defined in order to fairly compare the effectiveness showed by the different feature selection methods. The measure presented is an index of success *suc.*; see equation (3.1), which attempts to reward the selection of relevant features and to penalize the inclusion of irrelevant ones, in two situations:

- The solution is *incomplete*: there are relevant features lacking.

- The solution is *incorrect*: there are some irrelevant features.

$$suc. = \left[\frac{R_s}{R_t} - \alpha\frac{I_s}{I_t}\right] \times 100, \tag{3.1}$$

where R_s is the number of relevant features selected, R_t is the total number of relevant features, I_s is the number of irrelevant features selected and I_t is the total number of irrelevant features. The term α was introduced to question whether choosing an irrelevant feature is better than missing a relevant one (i.e., we prefer an incorrect solution rather than an incomplete one). The parameter α is defined as $\alpha = \min\{\frac{1}{2}, \frac{R_t}{I_t}\}$. Note that the higher the success, the better the method, and 100 is the maximum value.

In the case of ranker methods and in order to be fair, if all the optimal features are selected before any irrelevant feature, the index of success will be 100, due to the fact that the number of features that ranker methods are forced to select is always larger than the number of relevant features.

As was explained above, the evaluation of the feature selection methods is done by counting the number of correct/wrong features. However, it is also interesting and common practice in the literature [42] to see the classification accuracy obtained in a ten-fold cross-validation, in order to check if the true model (that is, the one with an index of success of 100) is also unique (that is, if it is the only one that can achive the best percentage of classification success). For this purpose, four well-known classifiers, based on different models, were chosen: C4.5, naive Bayes (NB), *k*-NN and SVM (see Appendix A). Experimental evidence has shown that decision trees, such as C4.5, exhibit a degradation in the performance when faced with many irrelevant features. Similarly, instance-based learners, such as *k*-NN, are also very susceptible to irrelevant features. It has been shown that the number of training instances needed to produce a predetermined level of performance for instance-based learning increases exponentially with the number of irrelevant features present [43]. On the other hand, algorithms such as naive Bayes are robust with respect to irrelevant features, degrading their performance very slowly when more irrelevant features are added. However, the performance of such algorithms deteriorate quickly by adding redundant features, even if they are relevant to the concept [44]. Finally, SVM can indeed suffer in high-dimensional spaces where many features are irrelevant [45].

3.3 Experimental Results

In all tables of this section, the best index of success and the best accuracy for each classifier are highlighted in boldface. Columns "Rel." (relevant), "Irr." (irrelevant) and "suc." (success, see equation (3.1)) refer to the evaluation via counting the number of correct features selected, whilst the remaining columns show the classification accuracy obtained by four different classifiers. It has to be noted that for the calculation of the index of success, the redundant attributes selected have the same penalization as the irrelevant features.

3.3.1 Dealing with Correlation and Redundancy: CorrAL

Two versions of this well-known datasets were used: CorrAL (the classic dataset) and CorrAL-100, formed by adding 93 irrelevant binary features (see Appendix A, Section A.2.2.1). The desired behavior of a feature selection method is to select the four relevant features and to discard the irrelevant ones, while detecting the correlated feature and not select it.

Table 3.1: Results for CorrAL. "C" indicates if the correlated feature is selected (✓) or not (✗)

Method	Rel.	C	Irr.	suc.	Accuracy (%)			
					C4.5	NB	k-NN	SVM
CFS	–	✓	0	-25	75.00	75.00	59.38	75.00
Consistency	–	✓	0	-25	75.00	75.00	59.38	75.00
INTERACT	–	✓	0	-25	75.00	75.00	59.38	75.00
InfoGain	–	✓	0	-25	75.00	75.00	59.38	75.00
ReliefF	1-4	✓	0	75	62.50	**81.25**	96.88	**87.50**
mRMR	1-4	✓	0	75	62.50	**81.25**	96.88	**87.50**
$M_d(\lambda = 0)$	1-4	✓	0	75	62.50	**81.25**	96.88	**87.50**
$M_d(\lambda = 1)$	1-4	✓	0	75	62.50	**81.25**	96.88	**87.50**
SVM-RFE	1-4	✓	0	75	62.50	**81.25**	96.88	**87.50**
SVM-RFE-G	1-4	✗	1	75	**81.25**	78.13	81.25	71.86
FS-P	1-4	✗	0	**100**	**81.25**	78.13	**100.00**	81.25
Wrapper SVM	–	✓	0	-25	75.00	75.00	59.38	75.00
Wrapper C4.5	–	✓	0	-25	75.00	75.00	59.38	75.00

Tables 3.1 and 3.2 show the results obtained over the datasets Corral and Corral-100, respectively, where the best results are highlighted in bold. Over Corral, FS-P was able to select the desired set of features, which led to 100% classification accuracy obtained by the k-NN classifier. Regarding Corral-100, it is curious that the best classification accuracy was obtained by SVM-RFE, which has an index of success of 25, which, however, can be explained as there are some irrelevant features in this dataset that are informative to the classifiers. This fact will be further analyzed in Section 3.5.

Table 3.2: Results for CorrAL-100. "C" indicates if the correlated feature is selected (✓) or not (✗)

Method	Rel.	C	Irr.	suc.	Accuracy (%)			
					C4.5	NB	k-NN	SVM
CFS	–	✓	0	-2	75.00	75.00	59.38	75.00
Consistency	–	✓	0	-2	75.00	75.00	59.38	75.00
INTERACT	–	✓	0	-2	75.00	75.00	59.38	75.00
InfoGain	–	✓	0	-2	75.00	75.00	59.38	75.00
ReliefF	1-3	✓	6	75	53.13	84.38	87.50	81.25
mRMR	1-4	✓	5	**99**	53.13	81.25	**90.63**	90.63
$M_d(\lambda = 0)$	1-4	✓	5	**99**	65.63	81.25	87.50	81.25
$M_d(\lambda = 1)$	1-4	✓	5	**99**	59.38	84.38	81.25	87.50
SVM-RFE	4	✓	8	25	62.50	**87.50**	68.75	**96.88**
SVM-RFE-G	–	✓	9	-44	68.75	68.75	62.50	75.00
FS-P	1,3,4	✓	6	75	53.13	**87.50**	84.38	87.50
Wrapper SVM	–	✓	0	-2	75.00	75.00	59.38	75.00
Wrapper C4.5	–	✓	2	-13	**84.38**	75.00	75.00	75.00

3.3.2 Dealing with Nonlinearity: XOR and Parity

In this subsection, the ability of feature selection methods to deal with relevance, irrelevance and redundancy will be checked over two nonlinear scenarios, XOR and Parity (see Appendix A, Sections A.2.2.2 and A.2.2.3). XOR-100 contains two relevant features and 97 irrelevant features whilst Parity3+3 has three relevant, three redundant and six irrelevant features. In the case of XOR-100, there is an added handicap of a small ratio between the number of samples and the number of features. For the sake of completeness, SVM and naive Bayes will be applied over these two datasets. However, bear in mind that those methods cannot solve nonlinear problems (SVM uses a linear kernel) and thus good results are not to be expected; so they will not be the focus of the analysis.

In Tables 3.3 and 3.4 it can be seen that the methods CFS, Consistency, INTERACT and InfoGain do not appear because they were not able to solve these nonlinear problems, returning an empty subset of features. On the other hand, the filter ReliefF and the embedded method SVM-RFE-G detected the relevant features both in XOR-100 and in Parity3+3, achieving the best indices of success and leading to high classification accuracies.

3.3.3 Dealing with Noise in the Inputs: Led

The Led dataset (see Appendix A, Section A.2.2.4) consists of correctly identifying seven leds that represent numbers between 0 and 9. Some irrelevant features were

Table 3.3: Results for XOR-100

Method	Rel.	Irr.	suc.	Accuracy (%)			
				C4.5	NB	k-NN	SVM
ReliefF	1,2	0	**100**	**100.00**	64.00	**100.00**	70.00
mRMR	1	9	50	52.00	74.00	64.00	72.00
$M_d(\lambda = 0)$	1	9	50	54.00	74.00	58.00	70.00
$M_d(\lambda = 1)$	1	9	50	58.00	70.00	62.00	62.00
SVM-RFE	–	10	-21	48.00	68.00	56.00	**78.00**
SVM-RFE-G	1,2	0	**100**	**100.00**	64.00	**100.00**	70.00
FS-P	1	9	50	62.00	**76.00**	62.00	74.00
Wrapper SVM	–	1	-2	66.00	66.00	60.00	66.00
Wrapper C4.5	1,2	2	99	**100.00**	70.00	96.00	50.00

Table 3.4: Results for Parity3+3

Method	Rel.	Red.	Irr.	suc.	Accuracy (%)			
					C4.5	NB	k-NN	SVM
ReliefF	1,2,3	2	0	93	**90.63**	29.69	**100.00**	37.50
mRMR	2,3	0	3	56	60.94	59.38	59.38	59.38
$M_d(\lambda = 0)$	–	0	5	-19	53.13	57.81	50.00	59.38
$M_d(\lambda = 1)$	–	0	5	-19	54.69	54.69	54.69	57.81
SVM-RFE	3	0	4	19	54.69	59.38	46.88	57.81
SVM-RFE-G	1,2,3	0	0	**100**	**90.63**	31.25	**100.00**	25.00
FS-P	–	0	5	-19	51.56	57.81	56.25	57.81
Wrapper SVM	–	0	1	-4	64.06	**64.06**	57.81	**64.06**
Wrapper C4.5	–	0	1	-4	64.06	**64.06**	57.81	**64.06**

added forming the Led-25 dataset (17 irrelevant features) and the Led-100 dataset (92 irrelevant attributes). In order to make these datasets more complex, different levels of noise in the inputs (2%, 6%, 10%, 15% and 20%) were added. It has to be noted that, as the attributes take binary values, adding noise means assigning an incorrect value to the relevant features.

Table 3.5: Results for Led-25 dataset with different levels of noise (N) in inputs

N(%)	Method	Relevant	Irr.	No. suc.	Accuracy (%) C4.5	NB	*k*-NN	SVM
0	CFS	1-5,7	0	86	92.00	**100.00**	**100.00**	96.00
	Consistency	1-5	0	71	**92.00**	**100.00**	**100.00**	96.00
	INTERACT	1-5,7	0	86	**92.00**	**100.00**	**100.00**	96.00
	InfoGain	1-7	0	**100**	**92.00**	**100.00**	**100.00**	96.00
	ReliefF	1-7	3	93	**92.00**	96.00	84.00	96.00
	mRMR	1-7	3	93	**92.00**	98.00	90.00	98.00
	$M_d(\lambda=0)$	1-5,7	4	76	90.00	92.00	82.00	98.00
	$M_d(\lambda=1)$	1-5,7	4	76	90.00	92.00	82.00	96.00
	SVM-RFE	1-7	3	93	**92.00**	98.00	80.00	96.00
	SVM-RFE n-l	1-7	0	**100**	**92.00**	**100.00**	**100.00**	96.00
	FS-P	1-7	0	**100**	**92.00**	**100.00**	**100.00**	96.00
	Wrapper SVM	1-5	2	67	**92.00**	90.00	82.00	**100.00**
	Wrapper C4.5	1-5	0	71	**92.00**	90.00	82.00	96.00
2	CFS	1-5	0	71	**90.00**	**98.00**	**96.00**	94.00
	Consistency	1-5	0	71	**90.00**	**98.00**	**96.00**	94.00
	INTERACT	1-5	0	71	**90.00**	**98.00**	**96.00**	94.00
	InfoGain	1-7	0	**100**	**90.00**	96.00	94.00	94.00
	ReliefF	1-7	3	93	86.00	88.00	82.00	88.00
	mRMR	1-7	3	93	84.00	88.00	82.00	88.00
	$M_d(\lambda=0)$	1-5,7	4	76	84.00	84.00	76.00	94.00
	$M_d(\lambda=1)$	1-5,7	4	76	84.00	84.00	78.00	88.00
	SVM-RFE	1-7	3	93	86.00	88.00	86.00	94.00
	SVM-RFE n-l	1-7	3	93	88.00	92.00	80.00	86.00
	FS-P	1-7	0	**100**	**90.00**	96.00	94.00	94.00
	Wrapper SVM	1-5	2	67	**90.00**	88.00	80.00	**96.00**
	Wrapper C4.5	1-5	0	71	**90.00**	**98.00**	**96.00**	94.00
6	CFS	1,2,4,5,7	0	71	70.00	**76.00**	**66.00**	68.00
	Consistency	1,2,4,5,7	0	71	70.00	**76.00**	**66.00**	68.00
	INTERACT	1,2,4,5,7	0	71	70.00	**76.00**	**66.00**	68.00
	InfoGain	1,2,4,5,7	0	71	70.00	**76.00**	**66.00**	68.00
	ReliefF	1-7	3	**93**	64.00	62.00	**66.00**	64.00
	mRMR	1-7	3	**93**	64.00	62.00	**66.00**	64.00
	$M_d(\lambda=0)$	1-5,7	4	76	62.00	62.00	60.00	68.00
	$M_d(\lambda=1)$	1-5,7	4	76	62.00	64.00	60.00	66.00
	SVM-RFE	1-4,7	5	59	62.00	62.00	58.00	68.00
	SVM-RFE-G	1-5	5	59	60.00	62.00	58.00	64.00
	FS-P	1-6	4	76	64.00	56.00	56.00	62.00
	Wrapper SVM	1,2,4,5	3	50	68.00	66.00	58.00	**70.00**
	Wrapper C4.5	1,2,4-6	1	69	**72.00**	68.00	58.00	66.00

Continued on next page

Table 3.5 – continued from previous page

N(%)	Method	Relevant	Irr.	No. suc.	Accuracy (%)			
					C4.5	NB	k-NN	SVM
10	CFS	1,2,4,7	0	57	60.00	50.00	**58.00**	46.00
	Consistency	1,2,4,7	0	57	60.00	50.00	**58.00**	46.00
	INTERACT	1,2,4,7	0	57	60.00	50.00	**58.00**	46.00
	InfoGain	1,2,4,7	0	57	60.00	50.00	**58.00**	46.00
	ReliefF	1-5,7	4	76	56.00	50.00	52.00	54.00
	mRMR	1-5,7	4	76	56.00	50.00	52.00	54.00
	$M_d(\lambda = 0)$	1-5,7	4	76	58.00	52.00	52.00	60.00
	$M_d(\lambda = 1)$	1-5,7	4	76	56.00	50.00	52.00	54.00
	SVM-RFE	2,4,7	7	26	44.00	40.00	42.00	50.00
	SVM-RFE n-1	1-5	5	59	58.00	48.00	46.00	56.00
	FS-P	1-7	3	**93**	60.00	48.00	**58.00**	52.00
	Wrapper SVM	1,2,4-6	2	67	66.00	54.00	52.00	**68.00**
	Wrapper C4.5	1-4	0	57	**68.00**	**64.00**	**58.00**	62.00
15	CFS	1,7	0	29	28.00	28.00	32.00	36.00
	Consistency	1,7	0	29	28.00	28.00	32.00	36.00
	INTERACT	1,7	0	29	28.00	28.00	32.00	36.00
	InfoGain	1,7	0	29	28.00	28.00	32.00	36.00
	ReliefF	1-7	3	**93**	48.00	38.00	46.00	52.00
	mRMR	1-7	3	**93**	48.00	38.00	46.00	52.00
	$M_d(\lambda = 0)$	1-5,7	4	76	48.00	42.00	44.00	52.00
	$M_d(\lambda = 1)$	1-5,7	4	76	44.00	36.00	46.00	48.00
	SVM-RFE	2,7	8	9	26.00	34.00	28.00	36.00
	SVM-RFE n-1	1,3,4,7	6	43	34.00	30.00	26.00	38.00
	FS-P	1-7	3	**93**	48.00	40.00	**48.00**	54.00
	Wrapper SVM	1,2	3	21	52.00	48.00	40.00	**56.00**
	Wrapper C4.5	1,2,5,7	0	57	**58.00**	44.00	40.00	54.00
20	CFS	1	0	14	28.00	20.00	28.00	28.00
	Consistency	1	0	14	28.00	20.00	28.00	28.00
	INTERACT	1	0	14	28.00	20.00	28.00	28.00
	InfoGain	1	0	14	28.00	20.00	28.00	28.00
	ReliefF	1-7	3	**93**	24.00	30.00	**40.00**	42.00
	mRMR	1-7	3	**93**	24.00	30.00	**40.00**	42.00
	$M_d(\lambda = 0)$	1-5,7	4	76	36.00	36.00	34.00	44.00
	$M_d(\lambda = 1)$	1-5,7	4	76	36.00	32.00	38.00	44.00
	SVM-RFE	5,6	8	9	16.00	24.00	18.00	20.00
	SVM-RFE n-1	1,2,4,5	6	43	38.00	36.00	20.00	34.00
	FS-P	1-7	3	**93**	34.00	34.00	36.00	40.00
	Wrapper SVM	1,2,5,7	3	50	44.00	38.00	38.00	**56.00**
	Wrapper C4.5	1,2,5	3	36	**48.00**	**40.00**	34.00	42.00

In Tables 3.5 and 3.6 one can see detailed results of these experiments. It is interesting to note that subset filters (CFS, Consistency and INTERACT) and the ranker filter Information Gain (which has a similar behavior to subset filters) do not select any of the irrelevant features in any case, at the expense of discarding some of the relevant ones, especially with high levels of noise. With regard to the classification accuracy, it decreases as the level of noise increases, as expected.

Table 3.6: Results for Led-100 dataset with different levels of noise (N) in inputs

N(%)	Method	Relevant	Irr.	No. suc.	Accuracy (%)			
					C4.5	NB	k-NN	SVM
0	CFS	1–5,7	0	86	**92.00**	**100.00**	**100.00**	96.00
	Consistency	1–5	0	71	**92.00**	**100.00**	**100.00**	96.00
	INTERACT	1–5,7	0	86	**92.00**	**100.00**	**100.00**	96.00
	InfoGain	1–7	0	**100**	**92.00**	**100.00**	**100.00**	96.00
	ReliefF	1–7	3	99	**92.00**	94.00	96.00	**100.00**
	mRMR	1–5,7	4	85	**92.00**	94.00	88.00	96.00
	$M_d(\lambda = 0)$	1–5,7	4	85	86.00	92.00	76.00	92.00
	$M_d(\lambda = 1)$	1–5,7	4	85	86.00	92.00	90.00	96.00
	SVM-RFE	3–7	5	71	46.00	54.00	48.00	48.00
	SVM-RFE-G	1–6	4	85	**92.00**	92.00	80.00	94.00
	FS-P	1–7	3	99	**92.00**	92.00	86.00	96.00
	Wrapper SVM	1–5	2	71	**92.00**	90.00	82.00	**100.00**
	Wrapper C4.5	1–5	0	71	**92.00**	**100.00**	**100.00**	96.00
2	CFS	1–5	0	71	**90.00**	**98.00**	**96.00**	94.00
	Consistency	1–5	0	71	**90.00**	**98.00**	**96.00**	94.00
	INTERACT	1–5	0	71	**90.00**	**98.00**	**96.00**	94.00
	InfoGain	1–7	0	**100**	**90.00**	96.00	94.00	94.00
	ReliefF	1–7	3	99	**90.00**	90.00	84.00	92.00
	mRMR	1–5,7	4	85	88.00	86.00	80.00	86.00
	$M_d(\lambda = 0)$	1–5,7	4	85	84.00	86.00	76.00	84.00
	$M_d(\lambda = 1)$	1–5,7	4	85	84.00	86.00	76.00	84.00
	SVM-RFE	3–7	5	71	68.00	70.00	54.00	70.00
	SVM-RFE-G	1–6	4	85	**90.00**	90.00	74.00	88.00
	FS-P	1–7	3	99	**90.00**	86.00	82.00	90.00
	Wrapper SVM	1–5	2	71	**90.00**	88.00	80.00	**96.00**
	Wrapper C4.5	1–5	0	71	**90.00**	**98.00**	**96.00**	94.00
6	CFS	1,2,4,5,7	0	71	72.00	**78.00**	**72.00**	70.00
	Consistency	1,2,4,5,7	0	71	72.00	**78.00**	**72.00**	70.00
	INTERACT	1,2,4,5,7	0	71	72.00	**78.00**	**72.00**	70.00
	InfoGain	1,2,4,5,7	0	71	72.00	**78.00**	**72.00**	70.00
	ReliefF	1–5,7	4	85	60.00	66.00	68.00	72.00
	mRMR	1–5,7	4	85	60.00	66.00	68.00	72.00
	$M_d(\lambda = 0)$	1,2,4,5,7	5	71	58.00	56.00	56.00	62.00
	$M_d(\lambda = 1)$	1–5,7	4	85	58.00	64.00	66.00	64.00
	SVM-RFE	2,3,5	7	42	52.00	50.00	34.00	52.00
	SVM-RFE-G	1–6	4	85	70.00	72.00	50.00	72.00
	FS-P	1–6	4	85	72.00	56.00	62.00	70.00
	Wrapper SVM	1–7	15	**99**	56.00	54.00	58.00	**84.00**
	Wrapper C4.5	1,2,4,5	2	57	**76.00**	72.00	66.00	72.00

Continued on next page

Table 3.6 – continued from previous page

N(%)	Method	Relevant	Irr.	No. suc.	Accuracy (%) C4.5	NB	k-NN	SVM
10	CFS	1,2,4,7	0	57	60.00	50.00	58.00	46.00
	Consistency	1,2,4,7	0	57	60.00	50.00	58.00	46.00
	INTERACT	1,2,4,7	0	57	60.00	50.00	58.00	46.00
	InfoGain	1,2,4,7	0	57	60.00	50.00	58.00	46.00
	ReliefF	1,2,4,5,7	5	71	74.00	54.00	66.00	64.00
	mRMR	1,2,4,5,7	5	71	66.00	**60.00**	66.00	66.00
	$M_d(\lambda = 0)$	1,2,4,5,7	5	71	68.00	58.00	**72.00**	60.00
	$M_d(\lambda = 1)$	1,2,4,5,7	5	71	74.00	56.00	62.00	66.00
	SVM-RFE	2,3,5,7	6	57	44.00	36.00	38.00	42.00
	SVM-RFE-G	1,3,5	7	42	26.00	34.00	30.00	40.00
	FS-P	1–6	4	**85**	60.00	46.00	48.00	58.00
	Wrapper SVM	1,2,4	9	42	72.00	56.00	56.00	**78.00**
	Wrapper C4.5	1,2,4	3	43	**76.00**	58.00	56.00	66.00
15	CFS	1,7	0	29	28.00	28.00	32.00	36.00
	Consistency	1,7	0	29	28.00	28.00	32.00	36.00
	INTERACT	1,7	0	29	28.00	28.00	32.00	36.00
	InfoGain	1,7	0	29	28.00	28.00	32.00	36.00
	ReliefF	1,2,4,5,7	5	**71**	54.00	**50.00**	**54.00**	**64.00**
	mRMR	1,2,4,5,7	5	**71**	54.00	**50.00**	**54.00**	**64.00**
	$M_d(\lambda = 0)$	1,2,4,5,7	5	**71**	54.00	**50.00**	**54.00**	**64.00**
	$M_d(\lambda = 1)$	1,2,4,5,7	5	**71**	**58.00**	**50.00**	52.00	56.00
	SVM-RFE	3,5,7	7	42	30.00	20.00	16.00	26.00
	SVM-RFE-G	1,5	8	28	16.00	24.00	12.00	16.00
	FS-P	1,3,5,6,7	5	**71**	30.00	28.00	22.00	26.00
	Wrapper SVM	1,2,6	5	42	50.00	**50.00**	42.00	**64.00**
	Wrapper C4.5	1,2,5,7	2	57	**58.00**	**50.00**	46.00	52.00
20	CFS	1	0	14	28.00	20.00	28.00	28.00
	Consistency	1	0	14	28.00	20.00	28.00	28.00
	INTERACT	1	0	14	28.00	20.00	28.00	28.00
	InfoGain	1	0	14	28.00	20.00	28.00	28.00
	ReliefF	1,2,5,7	6	57	30.00	**38.00**	**44.00**	44.00
	mRMR	1,2,5,7	6	57	34.00	**38.00**	42.00	**48.00**
	$M_d(\lambda = 0)$	1,2,5,7	6	57	32.00	**38.00**	38.00	32.00
	$M_d(\lambda = 1)$	1,2,5,7	6	57	38.00	32.00	28.00	34.00
	SVM-RFE	–	10	-1	8.00	26.00	20.00	20.00
	SVM-RFE-G	1,2,3,5	6	57	32.00	32.00	14.00	26.00
	FS-P	1–3,5,6	5	**71**	18.00	24.00	24.00	20.00
	Wrapper SVM	1	3	14	36.00	**38.00**	28.00	44.00
	Wrapper C4.5	1,5	4	28	**44.00**	32.00	28.00	32.00

3.3.4 Dealing with Noise in the Target: Monk3

In this subsection, the Monk3 problem, which includes 5% of misclassifications, i.e., noise in the target, will be tested. The relevant features are x_2, x_4 and x_5. However, as was stated by [46], for a feature selection algorithm it is easy to find the variables

x_2 and x_5, which together yield the second conjunction in the expression seen in Section A.2.2.5 (Appendix A). According to the experimental results presented by [46], selecting those features can lead to a better performance than selecting the three relevant ones. This additional information can help to explain the fact that in Table 3.7 several algorithms selected only two of the relevant features.

Table 3.7: Results for Monk3. Relevant features: 2, 4, 5

Method	Relevant	Irr.	No. suc.	Accuracy (%)			
				C4.5	NB	k-NN	SVM
CFS	2,5	0	67	**93.44**	88.52	89.34	79.51
Consistency	2,5	0	67	**93.44**	88.52	89.34	79.51
INTERACT	2,5	0	67	**93.44**	88.52	89.34	79.51
InfoGain	2,5	0	67	**93.44**	88.52	89.34	79.51
ReliefF	2,5,4	0	**100**	**93.44**	88.52	**90.98**	80.33
mRMR	2,5	3	17	92.62	88.52	80.33	78.69
$M_d(\lambda = 0)$	2,4,5	2	67	**93.44**	88.52	84.43	81.97
$M_d(\lambda = 1)$	2,4,5	2	67	**93.44**	88.52	84.43	81.97
SVM-RFE	2,4,5	2	67	**93.44**	88.52	84.43	**84.43**
SVM-RFE-G	2,4,5	2	67	**93.44**	88.52	84.43	**84.43**
FS-P	2,4,5	2	67	**93.44**	88.52	84.43	**84.43**
Wrapper SVM	2,4,5	1	83	**93.44**	**89.34**	82.79	79.51
Wrapper C4.5	2,5	0	67	**93.44**	88.52	89.34	79.51

Studying the index of success in Table 3.7, one can see that only ReliefF achieved a value of 100. The worst behavior was shown by mRMR, since it selected the three irrelevant features. As was justified above, many methods selected only two of the relevant features, which can be considered good comportment. For k-NN classifier, the best accuracy corresponds to ReliefF, which also obtained the best result in terms of index of success. Notice that C4.5 classifier achieves a very high classification accuracy (93.44%), especially bearing in mind that 5% of the samples are missclassifications and cannot be correctly identified.

3.3.4.1 Dealing with a Small Ratio Samples/Features: SD1, SD2 and SD3

These synthetic datasets (see Section A.2.2.6 in Appendix A) have a small ratio between number of samples and features, which makes difficult the task of feature selection, as explained in Section 1.3. This is the problem present in microarray data, a hard challenge for machine learning researchers. Besides these particularities of the data, there is a large number of irrelevant features for the task of gene classification, and also the presence of redundant variables is a critical issue.

Table 3.8: Features selected by each algorithm in synthetic dataset SD1

	(#)	OPT(2)	Red	Irr	suc	Accuracy (%)			
						C4.5	NB	k-NN	SVM
CFS	28	2	1	25	**100**	57.33	82.67	69.33	77.33
Cons	8	2	0	6	**100**	54.67	76.00	60.00	66.67
INT	23	2	0	21	**100**	60.00	81.33	66.67	80.00
IG	42	2	15	25	**100**	58.67	72.00	70.67	78.67
ReliefF[1]	2	1	1	0	50	40.00	45.33	44.00	46.67
ReliefF[2]	20	2	13	5	**100**	60.00	61.33	70.67	73.33
mRMR[1]	2	1	0	1	50	41.33	49.33	34.67	50.67
mRMR[2]	20	1	0	19	50	54.67	82.67	68.00	78.67
SVM-RFE[1]	2	2	0	0	**100**	56.00	60.00	52.00	57.33
SVM-RFE[2]	20	2	3	15	**100**	46.67	**88.00**	**76.00**	92.00
FS-P[1]	2	0	0	2	0	37.33	49.33	41.33	50.67
FS-P[2]	20	1	2	17	50	53.33	76.00	65.33	73.33
$\mathscr{M}_d(\lambda=0)$[1]	2	1	1	0	50	41.33	48.00	32.00	44.00
$\mathscr{M}_d(\lambda=0)$[2]	20	2	17	1	**100**	56.00	62.67	46.67	66.67
$\mathscr{M}_d(\lambda=1)$[1]	2	1	0	1	50	48.00	61.33	58.67	57.33
$\mathscr{M}_d(\lambda=1)$[2]	20	2	17	1	**100**	54.67	62.67	46.67	66.67
W-SVM	19	1	0	18	50	44.00	74.67	58.67	**94.67**
W-C4.5	10	0	0	10	0	**77.33**	38.67	40.00	38.67

[1] Selecting the optimal number of features.

[2] Selecting 20 features.

These datasets have groups of features that are all relevant to the class and redundant among them; so besides using the index of success and classification accuracy, we will use the measures employed in [47], which are more specific for this problem. Hence, the performance of SD1, SD2 and SD3 will be also evaluated in terms of:

- **(#)**: number of selected features.
- **OPT(x)**: number of selected features within the optimal subset, where x indicates the optimal number of features.
- **Red**: number of redundant features.
- **Irr**: number of irrelevant features.

For the ranker methods ReliefF, mRMR, $\mathscr{M}_d$, SVM-RFE and FS-P, two different cardinalities were tested: the optimal number of features and 20, since the subset methods have a similar cardinality. It has to be noted that in this problem and for the calculation of the index of success, redundant features are treated the same as irrelevant features in equation (3.1). Notice that the index of success is 100 even with 25 irrelevant features selected, due to the large number of irrelevant features (4,000).

Tables 3.8, 3.9 and 3.10 show the results for the datasets SD1, SD2 and SD3, respectively. Regarding the selected features, the subset filters and InfoGain (which

Table 3.9: Features selected by each algorithm in synthetic dataset SD2

	(#)	OPT(4)	Red	Irr	suc	Accuracy (%)			
						C4.5	NB	k-NN	SVM
CFS	21	4	0	17	**100**	64.00	**84.00**	72.00	81.33
Cons	9	4	0	5	**100**	54.67	70.67	60.00	72.00
INT	20	3	0	17	75	70.67	80.00	**74.67**	81.33
IG	40	4	19	17	**100**	61.33	69.33	61.33	76.00
ReliefF^1	4	0	0	4	0	48.00	64.00	50.67	52.00
ReliefF^2	20	1	9	10	25	54.67	60.00	61.33	70.67
mRMR^1	4	1	0	3	25	54.67	64.00	60.00	57.33
mRMR^2	20	1	0	19	25	60.00	70.67	44.00	68.00
SVM-RFE^1	4	3	1	0	75	46.67	62.67	54.67	65.33
SVM-RFE^2	20	4	4	12	**100**	57.33	82.67	69.33	**84.00**
FS-P^1	4	0	0	20	0	42.67	54.67	40.00	57.33
FS-P^2	20	0	0	20	0	52.00	68.00	42.67	61.33
$\mathscr{M}_d(\lambda = 0)^1$	4	2	2	0	50	56.00	56.00	26.67	50.67
$\mathscr{M}_d(\lambda = 0)^2$	20	4	16	0	**100**	54.67	64.00	49.33	68.00
$\mathscr{M}_d(\lambda = 1)^1$	4	1	0	3	25	46.67	69.33	56.00	69.33
$\mathscr{M}_d(\lambda = 1)^2$	20	1	9	10	25	52.00	62.67	60.00	74.67
W-SVM	13	1	0	12	25	44.00	60.00	45.33	77.33
W-C4.5	6	1	0	5	25	**72.00**	46.67	34.67	42.67

^{1} Selecting the optimal number of features.

^{2} Selecting 20 features.

exhibits similar behavior) showed excellent results, in all SD1, SD2 and SD3. Also SVM-RFE obtained good results, although the version with a Gaussian kernel could not been applied to these datasets due to memory complexity (notice that the simplest dataset, SD1, is formed by 4,020 features). With respect to the classifiers, SVM achieves the highest accuracies.

3.3.5 Dealing with a Complex Dataset: Madelon

Madelon (see Section A.2.2.7 in Appendix A) is a very complex artificial dataset which is distorted by adding noise, flipping labels, shifting and rescaling. It is also a nonlinear problem, so it provides a challenge for feature selection researchers [2]. The desired behavior for a feature selection method is for it to select the relevant features (1–5) and discard the redundant and irrelevant ones.

Table 3.11 shows the relevant features selected by the feature selection methods, as well as the number of redundant and irrelevant features elected by them and the classification accuracy. Notice that for the calculation of the success index, the redundant attributes selected stand for irrelevant features. Again, the results for SVM and naive Bayes will not be analyzed, since they are linear classifiers. The best re-

Table 3.10: Features selected by each algorithm in synthetic dataset SD3

	(#)	OPT(6)	Red	Irr	suc	Accuracy (%)			
						C4.5	NB	k-NN	SVM
CFS	23	4	2	17	**67**	64.00	80.00	**73.33**	70.67
Cons	9	3	0	6	50	58.67	76.00	62.67	76.00
INT	19	4	1	14	**67**	61.33	82.67	70.67	66.67
IG	49	4	31	14	**67**	62.67	65.33	65.33	73.33
ReliefF[1]	6	1	5	0	17	50.67	57.33	45.33	53.33
ReliefF[2]	20	1	9	10	17	56.00	69.33	61.33	68.00
mRMR[1]	6	1	0	5	17	62.67	62.67	66.67	65.33
mRMR[2]	20	1	0	19	17	50.67	77.33	52.00	66.67
SVM-RFE[1]	6	3	0	3	50	56.00	70.67	61.33	65.33
SVM-RFE[2]	20	4	2	14	**67**	49.33	**85.33**	70.67	**82.67**
FS-P[1]	6	0	0	6	0	36.00	54.67	34.67	46.67
FS-P[2]	20	1	0	19	17	38.67	61.33	45.33	56.00
$\mathscr{M}_d(\lambda = 0)$[1]	6	1	5	0	17	52.00	58.67	40.00	54.67
$\mathscr{M}_d(\lambda = 0)$[2]	20	3	15	2	50	45.33	57.33	50.67	54.67
$\mathscr{M}_d(\lambda = 1)$[1]	6	1	5	0	17	52.00	58.67	42.67	53.33
$\mathscr{M}_d(\lambda = 1)$[2]	20	1	9	10	17	54.67	66.67	60.00	70.67
W-SVM	10	1	0	9	17	48.00	61.33	61.33	81.33
W-C4.5	5	1	0	4	17	**68.00**	50.67	37.33	48.00

[1] Selecting the optimal number of features.

[2] Selecting 20 features.

sult in terms of success index was obtained by the wrapper with C4.5, selecting all the five relevant features, which also led to the best classification accuracy for C4.5. Notice that SVM-RFE with a Gaussian kernel could not be applied because of memory complexity.

3.4 Case Studies

After presenting the experimental results, and before discussing and analyzing them in detail, we will describe several case studies in order to decide among similar methods. These case studies will be based on the success index, to make this analysis classifier-independent.

3.4.1 Case Study I: Different Kernels for SVM-RFE

Two different kernels were applied on the embedded method SVM-RFE. The Gaussian kernel allows nonlinear problems to be solved, but at the expense of being

Table 3.11: Results for Madelon. Relevant features: 1–5

Method	Relevant	Red. No.	Irr. No.	suc.	Accuracy (%)			
					C4.5	NB	k-NN	SVM
CFS	3	7	0	20	80.92	69.58	86.83	66.08
Consistency	3,4	10	0	40	83.54	69.67	**90.83**	66.83
INTERACT	3,4	10	0	40	83.54	69.67	**90.83**	66.83
InfoGain	3,4	10	0	40	83.54	69.67	**90.83**	66.83
ReliefF	1,3,4,5	11	0	80	84.21	69.83	89.88	66.46
mRMR	–	1	14	0	64.92	62.25	53.13	57.08
$M_d(\lambda = 0)$	3,4	10	2	40	83.23	**70.21**	85.29	66.42
$M_d(\lambda = 1)$	3,4	10	2	40	83.23	**70.21**	85.29	66.42
SVM-RFE	1,3,4,5	4	7	80	86.42	66.88	81.25	67.42
FS-P	3,4	3	10	40	70.50	66.17	62.54	66.96
Wrapper SVM	3	0	16	20	66.63	66.04	54.08	**67.54**
Wrapper C4.5	1–5	5	15	**99**	**87.04**	70.00	75.42	66.33

more computationally demanding. In fact, SVM-RFE-G could not be applied to the datasets SD and Madelon, due to the space complexity. In Figure 3.1 one can see a comparison of these two versions of the method. Note that as for both Led-25 and Led-100 datasets there are results for six different levels of noise in the inputs, we have opted for computing the average of the index of success.

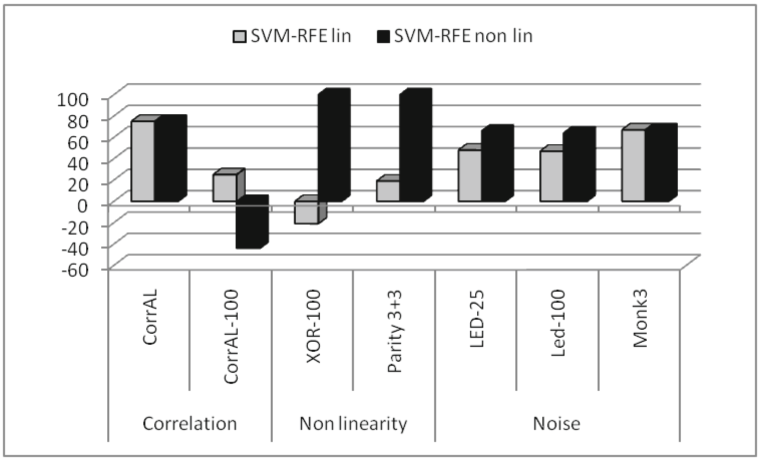


Fig. 3.1: SVM-RFE: Linear vs. Gaussian kernel. The vertical axis represents the success index

As expected, the linear kernel is not able to deal with nonlinear problems (XOR-100 and Parity3+3). On the other hand, the Gaussian kernel achieves a poor result over the Corral-100 dataset (where the number of irrelevant features increases considerably). In the remaining datasets, the Gaussian kernel maintains or increases the performance of the linear kernel. In these cases, it is necessary to bear in mind that the Gaussian kernel raises the computational time requirements of the algorithms and it cannot be applied over high-dimensional datasets (such as Madelon and the SD family). For example, over the XOR-100 dataset, the time used by the Gaussian kernel quadruplicates that used by the linear one. We suggest using the Gaussian kernel when there is some knowledge about the nonlinearity of the problem, and using the linear kernel in the remaining cases, especially when dealing with large amounts of data.

3.4.2 *Case Study II: mRMR vs $\mathscr{M}_d$*

The filter method $\mathscr{M}_d$ is an extension of mRMR which, instead of mutual information, uses a measure of dependence to assess relevance and irrelevance. Besides, it includes a free parameter (λ) that controls the relative emphasis given to relevance and irrelevance. In light of the above, the authors think that it is interesting to compare the behaviors shown by these two methods over the artificial datasets studied in this chapter. Two values of λ were tested, 0 and 1, and it is also important to see the difference between them. When λ is equal to 0, the effect of the redundancy disappears and the measure is based only on maximizing the relevance. On the other hand, when λ is equal to 1, it is more important to minimize the redundancy. For the sake of fairness, note that for the SD family of datasets, we considered the results achieved selecting 20 features.

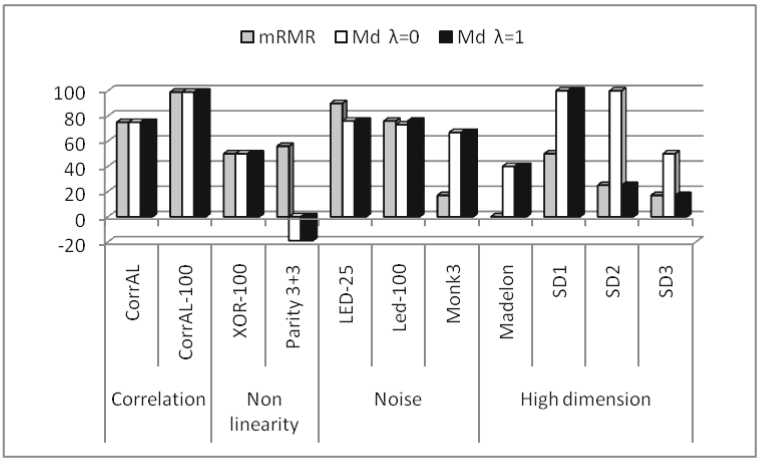


Fig. 3.2: mRMR *vs* $\mathscr{M}_d$. The vertical axis represents the success index

With regard to the different values of λ, one can see in Figure 3.2 that the index of success is the same for most of the datasets tested (eight out of 11). However, there is an important improvement in SD2 and SD3 when the value of λ is 0. Therefore, using this value of λ is recommended, although the appropriate value of λ is not an easy-to-solve question and requires further study and seems to be very dependent on the nature of data.

Comparing $\mathscr{M}_d$ and mRMR, the latter performs better in two datasets (Parity3+3 and Led-25) whereas $\mathscr{M}_d$ is better in five datasets (Monk3, Madelon, and the SD family). In the remaining datasets, the index of success achieved by both methods is the same. In light of these results, the use of $\mathscr{M}_d$ is recommended, except in datasets with high nonlinearity.

3.4.3 Case Study III: Subset Filters

Subset evaluation produces candidate feature subsets based on a certain search strategy. Each candidate subset is evaluated by a certain evaluation measure and compared with the previous best one with respect to this measure. This approach can handle feature redundancy together with feature relevance, despite releasing the user from the task of choosing how many features to retain.

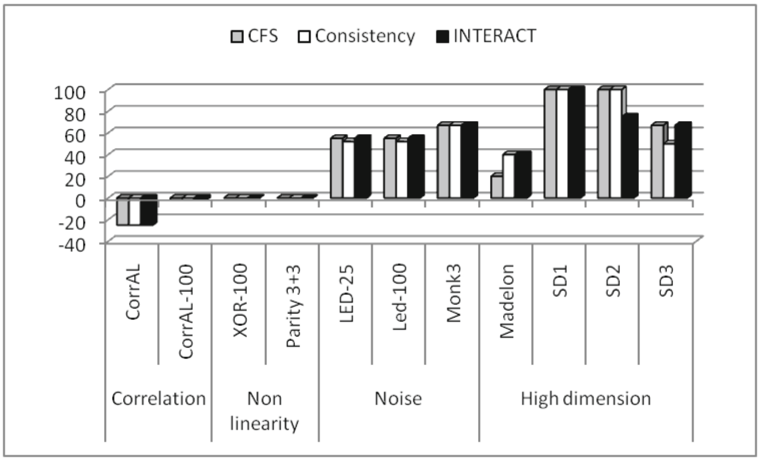


Fig. 3.3: Subset filters. The vertical axis represents the success index

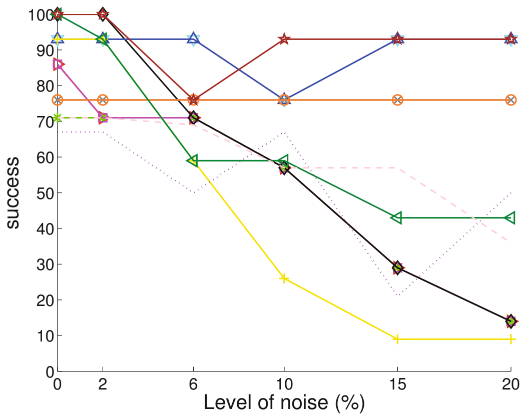
In Figure 3.3 one can see a comparison among the three subset filters considered in this study (CFS, INTERACT and Consistency-based) with regard to the index of success. In general all the three methods show very similar behavior, although some differences have been found. Consistency-based is slightly worse in datasets which present noise (Led-25, Led-100). This can be explained because, for this filter, a pattern is considered inconsistent if there are at least two instances which match all but their class labels, and therefore the given subset of features is inconsistent and the features are discarded. This case can happen when the data has been distorted with noise. On the other hand, CFS decays on Madelon. This method aims to maximize the correlation of the selected features with the class and to minimize the intercorrelation among the selected features. However, this algorithm cannot identify really strong interactions like the ones which may appear in parity problems (remember that Madelon is a generalization of a parity problem). In light of the above, using INTERACT is recommended.

3.4.4 Case Study IV: Different Levels of Noise in the Input

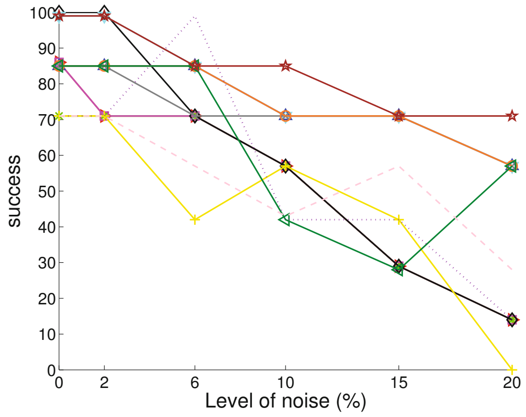
Figure 3.4 shows an overview of the behavior of feature selection methods with regard to different levels of noise, according to the success index described in (3.1). As we would have expected, in general the success index decreases when the level of noise increases, and worse performances were obtained over Led-100 due to the higher number of irrelevant features. It may seem strange that in some cases the success index improves with higher levels of noise (for example, in Led-100 from 10% to 15% of noise), but this fact can be explained by the random generation of the noise. Notice that the influence of each relevant feature is not the same in this problem, so adding noise to one or another may cause different results. In fact, the first five features (see Figure A.1 in Appendix A) are enough to distinguish among the ten digits; therefore if these attributes are distorted, the result may be altered.

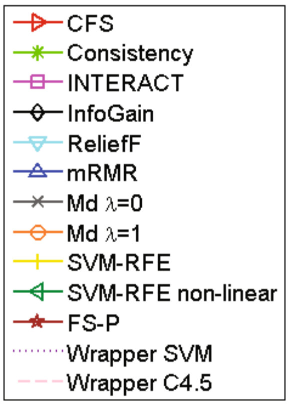
Several conclusions can be extracted from the graphs in Figure 3.4. Regarding the wrapper model, both versions tested degrade their results with the presence of noise, in both Led-25 and Led-100. With respect to embedded methods, two opposite behaviors have been observed. On the one hand, FS-P achieved very promising results on both datasets, without showing degradation as the level of noise increases. In fact, the index of success oscillates between 76 and 100 on Led-25—Figure 3.4a—and between 71 and 99 on Led-100—Figure 3.4b. On the other hand, the behavior of SVM-RFE (especially the version with the linear kernel) deteriorates considerably with high levels of noise. Note that SVM-RFE with linear kernel obtained 9 as success index on Led-25 and -1 on Led-100, which is the worst result for all the feature selection methods tested.

Concerning the filter model, mRMR and ReliefF are the methods that achieve the best success indices, ReliefF being slightly better in two cases (Led-100 with 0% and 2% of noise). These two filters obtain very good results without being very affected by noise. On the contrary, the subset filters (CFS, Consistency and INTERACT) and Information Gain are affected by high levels of noise although they are robust to the addition of irrelevant features. Finally, with respect to $\mathscr{M}_d$, it attains constant results, particularly on Led-25, and no significant differences have been found between the two values of λ tested.



(a) Led-25





(b) Led-100 (c) Legend

Fig. 3.4: Results for Led-25 and Led-100

The opposite behaviors of Information Gain and mRMR are curious, bearing in mind that both come from the Information Theory field. However, this fact can be explained because Information Gain is a univariate measure that considers the entropy between a given feature and the class level. On the other hand, mRMR takes into account the mutual information among features. The latter is a multivariate measure and therefore better behavior is expected when noise is present in data, because although some features may be affected by noise in a sample, not all of them are supposed to suffer it. This is why Information Gain obtains excellent results with low levels of noise but, as this increases, its performance decays until reaching a success index with a value of 14.

To sum up, the filters mRMR and ReliefF and the embedded method FS-P are the most tolerant to noise in the inputs and the subset filters (CFS, Consistency and INTERACT) and Information Gain are the most affected by noise.

3.5 Analysis and Discussion

In this section an analysis and discussion of the results presented in Section 3.3 will be carried out, to check which method is the best and to explain some behaviors shown in the experimental results. We will start by analyzing the success index and then we will discuss the relation between success index and classification accuracy, focusing on the specific problems studied in this chapter.

3.5.1 Analysis of Success Index

Table 3.12 shows the success average for each feature selection method over each scenario and also an overall average for each method (last column). For each Led-25 and Led-100 only one result is presented, respectively, corresponding to the average of the results for the distinct levels of noise tested. Analogously, for the SD family of datasets, only the average result of the three datasets is shown.

We are interested in an analysis of the success index (regardless of the classification accuracy) in order to check the behavior of the feature selection methods in a classifier-independent manner. In light of the results shown in Table 3.12, the best method according to the success index is the filter ReliefF, followed by the filters mRMR and both versions of $\mathscr{M}_d$. However, the subset filters and Information Gain showed poor results. Regarding the embedded model, FS-P is slightly better than SVM-RFE, and both of them are in the middle of the ranking. Finally, wrapper methods turned out to be the worst option in this study, since they achieved the poorest success averages.

Table 3.12: Average of success for every feature selection method tested. "N/A" stands for "Not Applicable"

	Correlation		Nonlinear		Noise			High dimension		
Method	Corr.	Corr100	XOR100	Par3+3	Led-25	Led-100	Monk3	SD	Mad.	Av.
CFS	-25	-2	0	0	55	55	67	89	20	29
Consistency	-25	-2	0	0	52	52	67	83	40	30
INTERACT	-25	-2	0	0	55	55	67	81	40	30
InfoGain	-25	-2	0	0	62	62	67	89	40	33
ReliefF	75	75	100	93	90	80	100	47	80	82
mRMR	75	99	50	56	90	76	17	31	0	55
M_d[1]	75	99	50	-19	76	73	67	83	40	60
M_d[2]	75	99	50	-19	76	76	67	47	40	57
SVM-RFE	75	25	-21	19	48	47	67	89	80	48
SVM-RFE-G	75	-44	100	100	66	64	67	N/A	N/A	N/A
FS-P	100	75	-19	-19	93	85	67	22	40	49
Wrap. SVM	-25	-2	-4	-4	54	57	83	31	20	23
Wrap. C4.5	-25	-13	-4	-4	60	55	67	22	99	29

[1] $\lambda = 0$
[2] $\lambda = 1$

In light of the results presented in Table 3.12, some guidelines are suggested:

- In complete ignorance of the particulars of data, the authors suggest using the filter ReliefF. It detects relevance in a satisfactory manner, even in complex datasets such as XOR-100, and it is tolerant to noise (both in the inputs and in the output). Moreover, due to the fact that it is a filter, it has the implicit advantage of its low computational cost.
- When dealing with high nonlinearity in data (such as XOR-100 and Parity3+3), SVM-RFE with a Gaussian kernel is an excellent choice, since it is able to solve these complex problems, however, at the expense of being computationally more expensive than the remaining approaches seen in these experiments.
- In the presence of altered inputs, the best option is to use the embedded method FS-P, since it has proved to be very robust to noise. A less expensive alternative is the use of the filters ReliefF or mRMR, which also show good behavior in this scenario. With low levels of noise (up to 6%), the use of the filter Information Gain is also recommended.
- When the goal is to select the smallest number of irrelevant features (even at the expense of selecting fewer relevant features), we suggest employing one of the subset filters (CFS, Consistency-based or INTERACT). These kinds of methods have the advantage of releasing the user from the task of deciding how many features to choose.
- When dealing with datasets with a small ratio between number of samples and features and a large number of irrelevant attributes, which is part of the problematics of microarray data, the subset filters and Information Gain presented promising behavior. SVM-RFE also performs adequately, but because of be-

ing an embedded method it is computationally expensive, especially in high-dimensional datasets like these.

- In general, the authors suggest the use of filters (specifically ReliefF), since they carry out the feature selection process independently of the induction algorithm and are faster than embedded and wrapper methods. However, in case of using another approach, we suggest using the embedded method FS-P.

As was stated in Chapter 2, filters are the methods with the lower computational cost whilst wrappers are computationally the most expensive. To illustrate this fact, in Figure 3.5 one can see the execution time of the different feature selection methods over two datasets: XOR-100 and Led-100 (with no noise). In order to be fair, mRMR was not included in this study because it was executed in a different machine, however one can expect a computational time similar to the one required by $\mathscr{M}_d$.

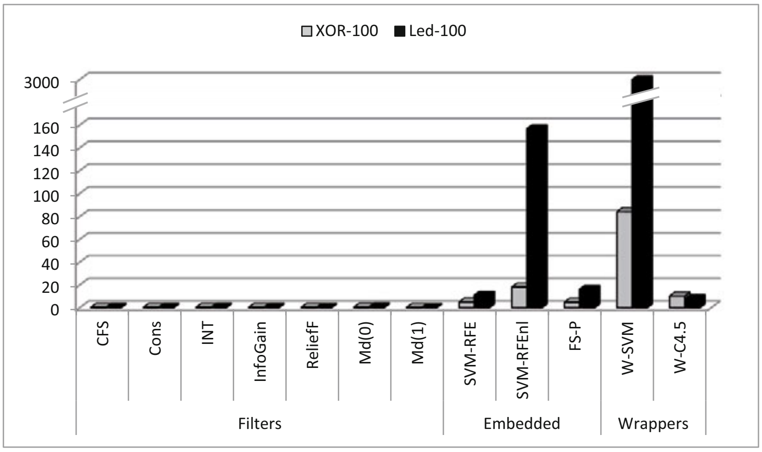


Fig. 3.5: Time in seconds for the datasets XOR-100 and Led-100

As expected, the filter model achieves the lowest execution times, always below one second. The embedded methods require more computational time, especially SVM-RFE with a Gaussian kernel. The time required by this method over Led-100 is especially high because this is a multiclass dataset, a fact that also increases the computational time. Wrapper SVM (which uses a linear kernel), contrary to Wrapper C4.5, is a very demanding method, particularly with Led-100, which needed almost one hour.

3.5.2 Analysis of Classification Accuracy

Although the previous analysis is interesting because it is not classifier-dependent, one may want to see the classification accuracy in order to check if the desired subset of features is unique and if it is the best option. For the sake of brevity, only the two extreme values of noise for the Led-25 and Led-100 datasets were included in this study. Table 3.13 shows, for each classifier and dataset, the best accuracy obtained, as well as the corresponding index of success and feature selection method employed. In this manner, it is easy to see at a glance if the best accuracy matches the best success index. In fact, this happens for all datasets except Led-100 with 20% of noise, where the inputs are clearly disturbed. This may be explained because the irrelevant features (randomly generated) are adding some useful information to the classifier, whilst the disturbed relevant features are not so informative.

The *k*-NN classifier, based on nearest neighbors, seems to be the best match for the proposed success index, since it obtains the best result for classification when the index also obtains its best result, specifically in five out of 13 datasets tested. Instance-based learners are very susceptible to irrelevant features; therefore when a feature selection method only selects the relevant features, its success index is high and is also the classification accuracy obtained by this classifier. It has to be also noted that *k*-NN is a nonlinear classifier; therefore it is capable of correctly classifying problems such as XOR-100 or Parity3+3, achieving 100% of classification accuracy when other methods like SVM obtained poor results.

SVM obtained the highest classification accuracy in seven out of 13 datasets shown in Table 3.13; however, it only coincides with the highest index of success in the SD2 dataset. This predictor takes advantage of the embedded method SVM-RFE and Wrapper SVM, both methods using this classifier performance to select the features. In fact, the highest accuracies were obtained after applying one of those methods for all datasets except for Led-25 with 20% of noise.

Although the behavior of the classifiers is interesting, one may want to focus on the problems studied in this chapter. For dealing with correlation and redundancy, two datasets were evaluated: CorrAL and CorrAL-100 (see Tables 3.1 and 3.2). Focusing on Corral, the subset filters, InfoGain and the wrappers selected only the correlated feature, which leads to an accuracy of 75% for all the classifiers except *k*-NN, which apparently is not able to take advantage of the relation between this feature and the class. When the four relevant features plus the correlated one are selected (rows 6–10 in Tables 3.1 and 3.2), one can see that the correlated feature (since it is correlated for 75% of data) is hindering the classification process, preventing the predictors from correctly classifying all samples. FS-P was the only method that selected the four relevant features and discarded the irrelevant and correlated ones; *k*-NN was able to achieve 100% classification accuracy, whilst the other methods were not. This fact is explained because of the complexity of the problem that may cause a given classifier to not solve the problem satisfactorily, even with the proper features. Regarding Corral-100, the highest accuracy (96.88%) was obtained by SVM having only one of the relevant features, the correlated one, and eight irrelevant ones. This fact can seem surprising but it can be explained because the irrelevant features (randomly generated) are informative in this problem. Classifying only with the relevant features and the correlated one, SVM achieves 65.62% classification accuracy; therefore it is clear that the irrelevant features are adding some useful information to the learner. In fact, by randomly generating 94 binary features and having only 32 samples, the probability that some of these irrelevant features could be correlated with the class is very high. This situation happens again with Wrapper C4.5 and C4.5 classifier, whilst the remaining methods exhibit similar behavior to Corral.

Nonlinearity is a difficult problem to deal with. In fact, two of the classifiers employed in this work (SVM with linear kernel and naive Bayes) and several feature selection methods do not turn very good results. This problem is present in XOR-100 and Parity3+3 datasets, in Tables 3.3 and 3.4. As we have said, naive Bayes and SVM cannot deal with nonlinearity; therefore they will not be the focus in this section. On the other hand, *k*-NN (and C4.5 only over XOR-100) achieve 100% classification accuracy when the desired features are selected. Over XOR-100, C4.5 obtains also 100% prediction accuracy after applying its own wrapper, even when it selected two extra irrelevant features. It has to be noted that this classifier performs an embedded selection of the features; therefore it may be using a smaller subset of features than the one given by the feature selection method. Finally, the improvement in SVM-RFE when using a Gaussian kernel for these kinds of datasets should be noted. SVM-RFE over XOR-100 did not select any of the relevant features, which led to a classification accuracy below the baseline accuracy. However, the computational time when using a Gaussian kernel is almost four times the one needed using a linear one (see Table 3.5), thus preventing its use in some of the datasets present in the study; so when to choose one or the other should be taken into account.

Different levels of noise in the input features were tested over the Led-25 (Table 3.5) and Led-100 (Table 3.6) datasets. As expected, the classification accuracy decreases when the level of noise increases. It has to be noted that, for both datasets, selecting five out of the seven relevant features is enough to achieve 100% classification accuracy. This happens because segments 1–5 (see Figure A.1, Appendix A) are enough to distinguish the ten digits (actually, five binary features allow 32 different states to be represented). In fact, when the level of noise is 6%, the first four methods miss the third feature (which allows digits 5 and 6 to be distinguished) and the performance declines 24%, which cannot be ascribed to the level of noise. This is a classification case that shows that the true model is not unique. On the other hand, it is curious that in some cases such as ReliefF over Led-25 with 20% noise, where it achieves a success index of 93 (selecting the seven relevant features), the maximum classification accuracy obtained with these features was 40% (SVM), which was not the expected result. This fact can be explained because of the high level of noise, which corrupts the relevant features and makes the classification task very complex. In those cases with high levels of noise, wrappers appear to be a good alternative, since they are classifier-dependent and try to search for the best features for the given classifier. To sum up, the filters mRMR and ReliefF and the embedded method FS-P are the methods that are most tolerant to noise in the inputs and the subsets filters (CFS, Consistency and INTERACT) and Information Gain are the most affected by noise, although wrappers are also a good choice if one is interested in maximizing the classification accuracy.

Table 3.13: Summary of results grouped by classifier. ‘Rf’ stands for ‘ReliefF’, ‘SR’ for ‘SVM-RFE’, ‘SRG’ for ‘SVM-RFE-G’, and ‘IG’ for ‘Information Gain’, respectively

	C4.5			NB			k-NN			SVM		
Dataset	Best Acc.	suc.	Method	Best Acc.	suc.	Method	Best Acc.	suc.	Method	Best Acc.	suc.	Method
CorrAL	81.25	**100**	FS-P	81.25	75	Rf,mRMR,M_d,SR	**100.00**	**100**	FS-P	87.50	75	Rf,mRMR,M_d,SR
		75	SRG									
CorrAL-100	84.38	-13	W-C4.5	87.50	75	FS-P	90.63	**99**	mRMR	**96.88**	25	SR
					25	SR						
XOR-100	**100.00**	**100**	Rf,SRG	76.00	50	FS-P	**100.00**	**100**	Rf,SRG	78.00	-21	SR
		99	W-C4.5									
Parity3+3	90.63	**100**	SRG	64.06	-4	W-SVM,W-C4.5	**100.00**	**100**	SRG	64.06	-4	W-SVM,W-C4.5
		93	Rf					93	Rf			
Led-25 (0%)	92.00	**100**	IG,SRG,FS-P	**100.00**	**100**	IG,SRG,FS-P	**100.00**	**100**	IG,SRG,FS-P	**100.00**	67	W-SVM
		93	Rf,mRMR,SR		86	CFS,INT		86	CFS,INT			
		86	CFS,INT		71	Cons		71	Cons			
		71	Cons,W-C4.5									
		67	W-SVM									
Led-25 (20%)	48.00	36	W-C4.5	40.00	36	W-C4.5	40.00	**93**	Rf,mRMR	**56.00**	50	W-SVM
Led-100 (0%)	92.00	**100**	IG	**100.00**	**100**	IG	**100.00**	**100**	IG	**100.00**	99	Rf
		99	Rf,FS-P		86	CFS,INT		86	CFS,INT		71	W-SVM
		86	CFS,INT		71	Cons,W-C4.5		71	Cons,W-C4.5			
		85	mRMR,SRG									
		71	Cons,W-SVM,W-C4.5									
Led-100 (20%)	44.00	28	W-C4.5	38.00	**57**[*]	Rf,mRMR,$M_d(0)$	44.00	**57**[*]	Rf	**48.00**	**57**[*]	mRMR
					14	W-SVM						
Monk3	**93.44**	**100**	Rf	89.34	83	W-SVM	90.98	**100**	Rf	84.43	67	SR
		83	W-SVM									SRG
		67	Rest except mRMR									FS-P
SD1	77.33	0	W-C4.5	88.00	**100**	SRG	76.00	**100**	SRG	**94.67**	50	W-SVM
SD2	72.00	25	W-C4.5	**84.00**	**100**	CFS	74.67	75	INT	**84.00**	**100**	SRG
SD3	68.00	17	W-C4.5	**85.33**	**67**	SRG	73.33	**67**	CFS	82.67	**67**	SRG
Madelon	87.04	**99**	W-C4.5	70.21	40	M_d	**90.83**	40	Cons,INT,IG	67.54	20	W-SVM

[*] This is not the highest index of success achieved.

The Monk3 dataset (see Table 3.7) is studied to deal with noise in the target. As was explained in Section 3.3.4, there is evidence that features x_2 and x_5 are enough for certain classifiers, which in fact happens in the experiments presented in this work. This is an example of the optimal feature subset being different than the subset of relevant features. On the other hand, again one can see the implicit capacity of C4.5 to select features, since it achieves the same result in cases where different subsets of features were selected, although for mRMR some of the irrelevant features caused the incorrect classification of one extra feature. This is not the case for k-NN classifier, which achieves the highest accuracy only when the “known” set of relevant features is selected. Naive Bayes and SVM seem to be more affected by misclassifications, since they obtain the worst results and do not take advantage of the best success indices.

SD1, SD2 and SD3 (Tables 3.8, 3.9 and 3.10) introduce the problem of microarray data: a small ratio between number of samples and features and a large number of redundant and irrelevant features. In general, the classification results are poor, because these kinds of problems are very difficult to solve since the classifiers tend to overfit. Moreover, the accuracy decreases when the complexity of the dataset increases (SD3). The embedded method SVM-RFE achieves very good results, especially with the SVM classifier. CFS and INTERACT filters also work satisfactorily together with naive Bayes and *k*-NN classifiers. The small ratio between number of samples and features prevents the use of wrappers, which have the risk of overfitting due to the small sample size. In fact, one can see in Tables 3.8, 3.9 and 3.10 that the wrappers obtain high accuracies in conjunction with their corresponding classifiers, but the performance decreases when using other classifiers. Regarding the classifiers, SVM achieves good results, especially over SD1. SVMs have many mathematical properties that make them attractive for gene expression analysis, including their flexibility in choosing a similarity function, sparseness of solution when dealing with large datasets, the ability to handle large feature spaces, and the ability to identify outliers [48]. Naive Bayes also obtained high accuracies, especially over SD2 and SD3. This learner is robust with respect to irrelevant features, although it deteriorates quickly by adding redundant features. In fact, it obtains the best accuracies when a small number of redundant features are present.

Madelon (Table 3.11) is a complex dataset which includes noise, flipping labels and nonlinearity. Due to the latter, naive Bayes and SVM cannot obtain satisfactory results; so they will not be analyzed. C4.5 obtained its highest accuracy after applying its own wrapper, as expected. The behavior of *k*-NN is more surprising; it obtained the highest prediction accuracy after applying methods that achieve poor success indices. However, this fact can be explained because these methods selected a large number of redundant features. These redundant features were built by multiplying the useful features by a random matrix; therefore they are also informative.

Table 3.14 shows the behavior of the different feature selection methods over the different problems studied, where the larger the number of dots, the better the behavior. To decide which methods were the most suitable under a given situation, a trade-off was computed between success index and classification accuracy. In light of these results, ReliefF turned out to be the best option independent of the particulars of the data, with the added benefit that it is a filter, which is the model with the lowest computational cost. However, SVM-RFE-G showed outstanding results, although its computational time is prohibitive in some cases (in fact, it could not be applied over some datasets). Wrappers have proven to be an interesting choice in some domains; nevertheless, they must be applied together with their own classifiers and it should be noted that this is the model with the highest computational cost. In addition, Table 3.14 provides some guidelines for specific problems.

3.6 Summary

Feature selection has been an active and fruitful field of research in machine learning. Its importance is indisputable and it has proven effective in increasing predictive accuracy and reducing complexity of machine learning models. However, choosing the appropriate feature selection method for a given scenario is not an easy-to-solve question. In this chapter, a review of eleven feature selection methods applied over eleven synthetic datasets was presented, aimed at studying their performance with respect to several situations that can hinder the process of feature selection. The

Table 3.14: General guidelines for specific problems

Method	Correlation and redundancy	Nonlinearity	Noise Inputs	Noise Target	No. feat >> No. samples
CFS	●	●	●	●●●	●●●●
Consistency	●	●	●	●●●	●●
INTERACT	●	●	●	●●●	●●●
InfoGain	●	●	●	●●●	●●●
ReliefF	●●●●	●●●●●	●●●●●	●●●●●	●●
mRMR	●●●●	●●●	●●●●●	●●	●
$M_d(\lambda = 0)$	●●●●	●●	●●●	●●●	●●●
$M_d(\lambda = 1)$	●●●●	●●	●●●	●●●	●●●
SVM-RFE	●●●●	●	●	●●●●	●●●●●
SVM-RFE-G	●●●●	●●●●●	●●●	●●●●	–
FS-P	●●●●●	●●	●●●●	●●●●	●
Wrapper SVM	●	●	●●●	●●●●	●●
Wrapper C4.5	●●	●●●	●●●	●●●	●●●

suite of synthetic datasets chosen covers phenomena such as the presence of irrelevant and redundant features, noise in the data or interaction among attributes. A scenario with a small ratio between the number of samples and features where most of the features are irrelevant was also tested. It reflects the problem of datasets such as microarray data, a well-known and hard challenge in the machine learning field, where feature selection becomes indispensable.

Within the feature selection field, three major approaches were evaluated: filters, wrappers and embedded methods. To test the effectiveness of the studied methods, an evaluation measure was introduced to reward the selection of the relevant features and to penalize the inclusion of the irrelevant ones. Also, four classifiers were selected to measure the effectiveness of the selected features and to check if the true model was also unique.

Some cases of study were also presented in order to decide from methods that showed similar behavior and help to find their adequacy in different situations. Finally, some guidelines about the appropriateness of the different feature selection methods in diverse scenarios have been proposed.

So far, this book has described the fundamentals and state of the art of feature selection. The next chapters are dedicated to studying feature selection in depth in applications such as DNA microarray classification or the medical domain.

References

1. Guyon, I. and Elisseeff, A. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

2. Guyon, I., Gunn, S., Nikravesh, M. and Zadeh, L. *Feature Extraction: Foundations and Applications*. Springer, 2006.
3. Yang, Y. and Pedersen, J. O. A comparative study on feature selection in text categorization. In *International Conference on Machine Learning*, pages 412–420, 1997.
4. Yu, L. and Liu, H. Efficient feature selection via analysis of relevance and redundancy. *The Journal of Machine Learning Research*, 5:1205–1224, 2004.
5. Provost, F. Distributed data mining: Scaling up and beyond. *Advances in Distributed and Parallel Knowledge Discovery*, pages 3–27, 2000.
6. Yu, L. and Liu, H. Redundancy based feature selection for microarray data. In *10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 737–742. ACM, 2004.
7. Bolon-Canedo, V., Sanchez-Marono, N. and Alonso-Betanzos, A. Feature selection and classification in multiple class datasets: An application to KDD Cup 99 dataset. *Expert Systems with Applications*, 38(5):5947–5957, 2011.
8. Lee, W., Stolfo, S. J. and Mok, K. W. Adaptive intrusion detection: A data mining approach. *Artificial Intelligence Review*, 14(6):533–567, 2000.
9. Forman, G. An extensive empirical study of feature selection metrics for text classification. *The Journal of Machine Learning Research*, 3:1289–1305, 2003.
10. Gomez, J. C., Boiy, E. and Moens, M. F. Highly discriminative statistical features for email classification. *Knowledge and Information Systems*, 31(1):23–53, 2012.
11. Egozi, O., Gabrilovich, E. and Markovitch, S. Concept-based feature generation and selection for information retrieval. In *23rd AAAI Conference on Artificial Intelligence*, pages 1132–1137, 2008.
12. Dy, J. G., Brodley, C. E., Kak, A., Broderick, L. S. and Aisen, A. M. Unsupervised feature selection applied to content-based retrieval of lung images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(3):373–378, 2003.
13. Saari, P., Eerola, T. and Lartillot, O. Generalizability and simplicity as criteria in feature selection: application to mood classification in music. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(6):1802–1812, 2011.
14. Sun, Y., Babbs, C. F. and Delp, E. J. A comparison of feature selection methods for the detection of breast cancers in mammograms: adaptive sequential floating search vs. genetic algorithm. In *IEEE-EMBS 27th Annual International Conference of the Engineering in Medicine and Biology Society*, pages 6532–6535. IEEE, 2006.
15. Liu, H., Liu, L. and Zhang, H. Feature selection using mutual information: An experimental study. In *PRICAI 2008: Trends in Artificial Intelligence*, pages 235–246. Springer, 2008.
16. Beretta, L. and Santaniello, A. Implementing ReliefF filters to extract meaningful features from genetic lifetime datasets. *Journal of Biomedical Informatics*, 44(2):361–369, 2011.
17. Zhang, M. L., Peña, J. M. and Robles, V. Feature selection for multi-label naive Bayes classification. *Information Sciences*, 179(19):3218–3229, 2009.
18. Perner, P. and Apte, C. Empirical evaluation of feature subset selection based on a real-world data set. In *Principles of Data Mining and Knowledge Discovery*, volume 1910 of *Lecture Notes in Computer Science*, pages 575–580. Springer Berlin Heidelberg, 2000.
19. George, V. S. and Raj, V. C. Review on feature selection techniques and the impact of SVM for cancer classification using gene expression profile. *International Journal of Computer Science & Engineering Survey*, 2(3):16–27, 2011.
20. Chidlovskii, B. and Lecerf, L. Scalable feature selection for multi-class problems. In *Machine Learning and Knowledge Discovery in Databases*, pages 227–240. Springer, 2008.
21. Li, T., Zhang, C. and Ogihara, M. A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 20(15):2429–2437, 2004.
22. Hua, J., Tembe, W. D. and Dougherty, E. R. Performance of feature-selection methods in the classification of high-dimension data. *Pattern Recognition*, 42(3):409–424, 2009.
23. Bontempi, G. and Meyer, P. E. Causal filter selection in microarray data. In *27th International Conference on Machine Learning*, pages 95–102, 2010.

24. Aliferis, C. F., Statnikov, A., Tsamardinos, I., Mani, S. and Koutsoukos, X. D. Local causal and Markov blanket induction for causal discovery and feature selection for classification. Part I: Algorithms and empirical evaluation. *The Journal of Machine Learning Research*, 11:171–234, 2010.
25. Zhang, Y., Ding, C. and Li, T. Gene selection algorithm by combining ReliefF and mRMR. *BMC Genomics*, 9(Supl 2):S27, 2008.
26. Byeon, B. and Rasheed, K. Simultaneously removing noise and selecting relevant features for high dimensional noisy data. In *7th International Conference on Machine Learning and Applications*, pages 147–152. IEEE, 2008.
27. Yang, S. H. and Hu, B. G. Efficient feature selection in the presence of outliers and noises. In *Information Retrieval Technology*, pages 184–191. Springer, 2008.
28. Guyon, I., Bitter, H. M., Ahmed, Z., Brown, M. and Heller, J. Multivariate non-linear feature selection with kernel methods. In *Soft Computing for Information Processing and Analysis*, pages 313–326. Springer, 2005.
29. Liu, H. and Yu, L. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):491–502, 2005.
30. Brown, G., Pocock, A., Zhao, M.J. and Luján, M. Conditional likelihood maximisation: A unifying framework for information theoretic feature selection. *The Journal of Machine Learning Research*, 13(1): 27–66. 2012.
31. Vergara, J. R. and Estévez, P. A. A review of feature selection methods based on mutual information. *Neural Computing and Applications*, 24(1): 175–186. 2014.
32. Molina, L. C, Belanche, L. and Nebot, A. Feature selection algorithms: A survey and experimental evaluation. In *IEEE International Conference on Data Mining*, pages 306–313. IEEE, 2002.
33. Doak, J. *An Evaluation of Feature Selection Methods and Their Application to Computer Security*. University of California, Computer Science, 1992.
34. Jain, A. and Zongker, D. Feature selection: Evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–158, 1997.
35. Kudo, M. and Sklansky, J. A comparative evaluation of medium and large-scale feature selectors for pattern classifiers. In *1st International Workshop on Statistical Techniques in Pattern Recognition*, pages 91–96, 1997.
36. Liu, H. and Setiono, R. Scalable feature selection for large sized databases. In *4th World Conference on Machine Learning*, pages 101–106, 1998.
37. Thrun, S. B., Bala, J. W., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., De Jong, K. A., Dzeroski, S., Fisher, D. H., Fahlman, S. E. and others. The monk's problems a performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Carnegie Mellon University, 1991.
38. Belanche, L. A. and González, F. F. Review and evaluation of feature selection algorithms in synthetic problems. *arXiv preprint arXiv:1101.2320*, 2011.
39. Rakotomamonjy, A. Variable selection using SVM based criteria. *The Journal of Machine Learning Research*, 3:1357–1370, 2003.
40. Mejía-Lavalle, M., Sucar, E. and Arroyo, G. Feature selection with a perceptron neural net. In *International Workshop on Feature Selection for Data Mining*, pages 131–135, 2006.
41. Sánchez-Maroño, N., Alonso-Betanzos, A. and Tombilla-Sanromán, M. Filter methods for feature selection—a comparative study. In *Intelligent Data Engineering and Automated Learning-IDEAL 2007*, pages 178–187. Springer, 2007.
42. Mamitsuka, H. Query-learning-based iterative feature-subset selection for learning from high-dimensional data sets. *Knowledge and Information Systems*, 9(1):91–108, 2006.
43. Langley, P. and Iba, W. Average-case analysis of a nearest neighbor algorithm. In *International Joint Conference on Artificial Intelligence*, volume 13, pages 889–889. Citeseer, 1993.
44. Pradhananga, N. Effective linear-time feature selection. *Ph.D. Thesis*, 2007.
45. Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T. and Vapnik, V. Feature selection for SVMs. In *Neural Information Processing Systems Conference*, volume 12, pages 668–674, 2000.

46. Kohavi, R. and John, G. H. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1):273–324, 1997.
47. Zhu, Z., Ong, W. S. and Zurada, J. M. Identification of full and partial class relevant genes. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(2):263–277, 2010.
48. Brown, M. P. S., Grundy, W. N., Lin, D., Cristianini. N, Sugnet, C.W., Furey, T.S., Ares, M. and Haussler, D. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Sciences*, 97(1):262–267, 2000.

Chapter 4
Feature Selection in DNA Microarray Classification

Abstract Microarray data classification is a difficult challenge for machine learning researchers due to its large number of features and small sample sizes. Since its introduction, feature selection has been considered a de facto standard in the field, and a huge number of feature selection methods were utilized trying to reduce the input dimensionality while improving the classification performance. This chapter is devoted to reviewing the most up-to-date feature selection methods developed in this field. Section 4.1 introduces the background and first attempts to deal with this type of data. Next, Section 4.2 provides a description of the inherent problematics of microarray data, such as the small sample size, the imbalance of the data, the dataset shift or the presence of outliers. In Section 4.3 we review the state of the art on feature selection methods applied to this type of data. In Section 4.4 we present an experimental study of the most significant algorithms and evaluation techniques. A deep analysis of the findings of this study is also provided (Section 4.5). Finally, in Section 4.6, we summarize the contents of this chapter.

As mentioned in previous chapters, feature selection has been widely studied in recent years by machine learning researchers. This technique has found success in many different real-world applications; among them it is worth mentioning DNA microarray analysis, since it is an important representative of high-dimensional data.

Over the last two decades, the advent of DNA microarray datasets has stimulated a new line of research in bioinformatics and in machine learning. This type of data is used to collect information from tissue and cell samples regarding gene expression differences that could be useful for diagnosing disease or for distinguishing a specific tumor type. Although there are usually very few samples (often fewer than 100 patients) for training and testing, the number of features in the raw data ranges from 6,000 to 60,000, since it measures the gene expression en masse, and converts this problem to the classical example of reduced sample size illustrated in Chapter 1. A

Part of the content of this chapter was previously published in *Information Sciences* (`doi:10.1016/j.ins.2014.05.042`)

V. Bolón-Canedo et al., *Feature Selection for High-Dimensional Data*, Artificial Intelligence: Foundations, Theory, and Algorithms, DOI 10.1007/978-3-319-21858-8_4

typical classification task is to separate healthy patients from cancer patients based on their gene expression “profile” (binary approach). There are also datasets where the goal is to distinguish between different types of tumors (multiclass approach), making the task even more complicated.

Therefore, microarray data pose a great challenge for machine learning researchers. Having so many fields relative to so few samples creates a high likelihood of finding “false positives” that are due to chance (both in finding relevant genes and in building predictive models) [1]. It becomes necessary to find robust methods to validate the models and assess their likelihood. Furthermore, additional experimental complications like noise and variability render the analysis of microarray data an exciting domain [2].

Several studies have shown that most genes measured in a DNA microarray experiment are not relevant for an accurate classification among different classes of the problem [3]. To avoid the “curse of dimensionality” [4], feature (gene) selection plays a crucial role in DNA microarray analysis. It soon became indispensable among the researchers, not only to remove redundant and irrelevant features, but also to help biologists to identify the underlying mechanism that relates gene expression to diseases. This research area has received significant attention in recent years (most of the work has been published in the last decade), and new algorithms have emerged as alternatives to the existing ones. However, when a new method is proposed, there is a lack of standard state-of-the-art results to perform a fair comparative study. Besides, there is a broad suite of microarray datasets to be used in the experiments; some of them are even named the same, but the number of samples or characteristics are different in different studies, which makes this task more complicated.

The main goal of this chapter is to provide a framework for ongoing studies, paying attention to the datasets used in the experimental analysis, the intrinsic data characteristics and the experimental analysis of classical feature selection algorithms available in data mining software tools used for microarray data. In this manner, it is possible to be aware of the particularities of this type of data as well as its problems (some of them commented on in Chapter 1), such as the imbalance of the data, their complexity, the presence of overlapping and outliers, or the so-called dataset shift. These problems render the analysis of microarray data an exciting domain.

An experimental framework has been designed in such a way that well-founded conclusions can be extracted. A set of nine binary microarray datasets is used, which suffer from problems such as class imbalance, overlapping or dataset shift. Some of these datasets come originally divided into training and test datasets, so a holdout validation is performed on them. For the remaining datasets, they will be evaluated with a k-fold cross-validation, since it is a common choice in the literature. However, it has been shown that cross-validation can potentially introduce dataset shift, so another strategy has been included to create the partitioning, called *Distribution optimally balanced stratified cross-validation* (DOB-SCV) [5]. C4.5, Support Vector Machine (SVM) and naive Bayes were selected as classifiers, and we use classification accuracy, sensitivity and specificity on the test partitions as the evaluation criteria.

4.1 Background: The Problem and First Attempts

All cells have a nucleus, and inside the nucleus there is DNA, which encodes the "program" for future organisms. DNA has coding and non-coding segments. The coding segments, also known as genes, specify the structure of proteins, which do the essential work in every organism. Genes make proteins in two steps: DNA is transcribed into mRNA and then mRNA is translated into proteins. Advances in molecular genetics technologies, such as DNA microarrays, allow us to obtain a global view of the cell, where it is possible to measure the simultaneous expression of tens of thousands of genes [1]. Figure 4.1 displays the general process of acquiring the gene expression data from a DNA microarray. These gene expression profiles can be used as inputs to large-scale data analysis, for example, to increase our understanding of normal and disease states.

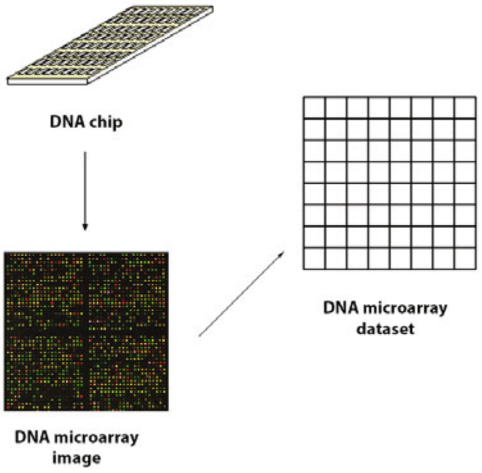


Fig. 4.1: General process of acquiring the gene expression data from DNA microarray

Microarray datasets began to be dealt with at the end of the 1990s. Soon feature (gene) selection was considered a de facto standard in this field. Further work was carried out in the beginning of the 2000s [2]. The univariate paradigm, which is fast and scalable but ignores feature dependencies, has dominated the field during the 2000s [6, 7, 8]. However, there were also attempts to tackle microarray data with multivariate methods, which are able to model feature dependencies, but at the cost of being slower and less scalable than univariate techniques. In addition to the application of multivariate filter methods [9, 10, 11, 12], the microarray problem was also treated with more complex techniques such as wrappers and embedded methods [13, 14, 15, 16].

4.2 Intrinsic Characteristics of Microarray Data

As mentioned at the beginning of this chapter, microarray data classification posed a great challenge for computational techniques, because of their large dimensionality (up to several tens of thousands of genes) and small sample sizes. Furthermore, additional experimental complications exist, as mentioned above, that render the analysis of microarray data an exciting domain.

4.2.1 Small Sample Size

The first problem that one may find when dealing with microarray data is related with the small samples size (usually less than 100). A key point in this situation is that error estimation is greatly impacted by small samples [17]. Without the appropriate estimation of the error, an unsound application of classification methods exists, which has generated a large number of publications and an equally large number of unsubstantiated scientific hypotheses [18]. For example, [19] reported that reanalysis of data from the seven largest published microarray-based studies that have attempted to predict prognosis of cancer patients reveals that five out of those seven did not classify patients better than by chance. To overcome this problem, it becomes necessary to select a correct validation method for estimating the classification error.

4.2.2 Class Imbalance

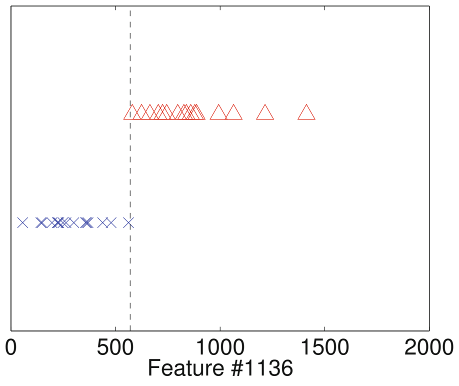
A common problem in microarray data is the so-called *class imbalance problem*. This occurs when a dataset is dominated by a major class or classes which have significantly more instances than the other rare or minority classes in the data [20, 21, 22]. Typically, people have more interest in learning rare classes. For example, in the domain at hand, the cancer class tends to be rarer than the non-cancer class because usually there are more healthy patients. However, it is important for practitioners to predict and prevent the apparition of cancer. In these cases, standard classifier learning algorithms have a bias toward the classes with a greater number of instances, since rules that correctly predict those instances are positively weighted in favor of the accuracy metric, whereas specific rules that predict examples from the minority class are usually ignored (treated as noise), because more general rules are preferred. Therefore, minority class instances are misclassified more often than those from the other classes [23]. Although class imbalance does not hinder the learning task by itself, there are some difficulties related to this problem that turn up, such as a small sample size, as in the case of microarray data. This problematic is of special importance when the imbalance is more marked in the test set than in the training set, as will be further discussed in Section 4.2.4 dealing with the dataset shift problem. Multiclass datasets also suffer this problem. For example, the widely used Lymphoma dataset [24] has nine classes but the majority class takes 48% of the samples. Traditional preprocessing techniques to overcome this issue are undersampling methods, which create a subset of the original dataset by eliminating instances; oversampling methods, which create a superset of the original dataset by replicating some instances or creating new instances from existing ones; and finally, hybrid methods that combine both sampling methods. One of the most employed resampling techniques is the so-called SMOTE [25], where the minority class is oversampled by taking each minority class sample and introducing synthetic examples along the line segments joining any or all of the *k* minority class nearest neighbors. This technique was applied by [26] on microarray data, although the authors stated that it does not attenuate the bias toward the classification in the majority class for most classifiers. In recent years, ensembles of classifiers have arisen as a possible solution to the class imbalance problem, attracting great interest among researchers [23, 27], in several cases combined with preprocessing techniques such as SMOTE. Ensemble-based algorithms have proven to improve the results that are obtained by the usage of data preprocessing techniques and training a single classifier. For all these reasons, it is worth considering this problematic when dealing with unbalanced microarray datasets.

4.2.3 Data Complexity

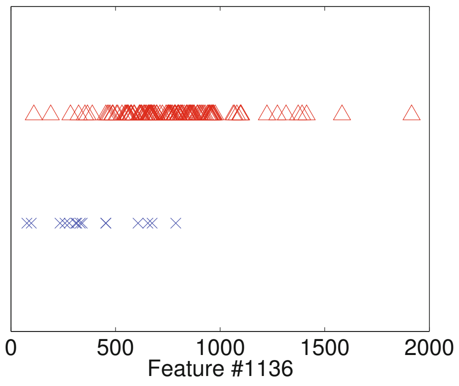
Data complexity measures are a recent proposal to represent characteristics of the data which are considered difficult in classification tasks, such as the overlapping among classes, their separability or the linearity of the decision boundaries [28, 29]. Particularly, these measures have been referred to as gene expression analysis by [30] and [31], demonstrating that low complexity corresponds to small classification error. In particular, the measures of *class overlapping*, such as F1 (*maximum Fisher's discriminant ratio*) [28], focus on the effectiveness of a single feature dimension in separating the classes. They examine the range and spread of values in the dataset within each class and check for overlapping among different classes.

4.2.4 Dataset Shift

Another common problem when datasets come originally divided in training and test sets is the so-called *dataset shift*. It occurs when the testing (unseen) data experience a phenomenon that leads to a change in the distribution of a single feature, a combination of features, or the class boundaries [32]. As a result, the common assumption that the training and testing data follow the same distributions is often violated in real-world applications and scenarios, which may hinder the process of feature selection and classification. For example, Lung [33] and Prostate [34] datasets have separated training and test sets. In the case of Lung, there is a single feature (#1136) which can correctly classify all the samples in the training set, as shown in Figure 4.2a, where different colors and shapes stand for different classes and the dashed line shows a clear linear separation between them. However, the same feature is not that informative in the test set and the class is not linearly separable, as displayed in Figure 4.2b. Besides, note that there is a enormous disparity in the class distribution: 50-50% in the training set vs. 90-10% in the test set.



(a) Lung Train



(b) Lung Test

Fig. 4.2: Feature #1136 in Lung dataset

The Prostate dataset poses a big challenge for machine learning methods since the test dataset was extracted from a different experiment and has a nearly 10-fold difference in overall microarray intensity from the training data. In fact, the test distribution (26-74%) differs significantly from the train distribution (49-51%), and with an inappropriate feature selection, some classifiers just assign all the samples to one of the classes [35, 36].

Dataset shift can also appear when performing a cross-validation technique, which divides the whole training set into several subsets of data. In this case, there are some partitioning methods which may solve the problem [5].

4.2.5 Outliers

An important aspect that has been neglected in the literature is to detect outliers [37] in microarray samples. In some microarray datasets, there are samples incorrectly labeled or identified as likely contaminated which should be designated outliers, since they can exert a negative effect on the selection of informative genes for sample classification. The authors in [38] developed a method which found some outlying samples in the well-known Colon dataset. Therefore, analysis of samples designated as outliers should be considered as a preprocessing step in classification of microarray datasets because they can have a negative effect in the gene subset selection and, as a consequence, in the final prediction [39].

4.3 Algorithms for Feature Selection on Microarray Data: A Review

Feature selection methods are constantly emerging and, for this reason, there is a wide suite of methods that deal with microarray gene data. The aim of this section is to present those methods developed in the last few years. Traditionally in the feature selection domain, the gene selection methods most employed fall into the filter approach (see Section 4.3.1). Most of the novel filter methods proposed are based on information theory, although issues such as robustness or division in multiple binary problems are emerging topics. Discretization as a step prior to feature selection has also received some attention. On the other hand, and due to the heavy computational consumption of resources and the high risk of overfitting, the wrapper approach has been mostly avoided in the literature (see Section 4.3.2). Although the embedded model had not received enough attention during the infancy of microarray data classification, several proposals appeared over the last few years, as reported in Section 4.3.3. It is also worth noticing that the review of the up-to-date literature showed a tendency to mix algorithms, in the form of either hybrid methods or ensemble methods. Also, it is well known that genes interact with each other through gene regulative networks; so clustering methods have also been proposed. These novel approaches will be described in Section 4.3.4.

4.3.1 Filters

Filter methods evaluate the goodness of gene subsets by observing only intrinsic data characteristics (i.e., statistical measures), where typically a single gene or a subset of genes is evaluated against the class label. Classical filter methods are usually applied to microarray data, such as Correlation Feature Selection (CFS), Fast Correlation-Based Filter (FCBF), ReliefF, or the consistency-based filter (see Chapter 2). Also, the well-known and widely used minimum Redundancy Maximum Relevance (mRMR) has proven to be an appropriate tool for gene selection. During the last lustrum, besides the application of known methods, an important number of filter approaches have been proposed and applied to microarray datasets, and this subsection will review the most interesting ones. An important number of filters are based on information theory, as can be seen in Section 4.3.1.1. On the other hand, several approaches include a preprocessing step to discretize data, since some filters require the data to be discrete, as reported in Section 4.3.1.2. Section 4.3.1.3 presents the filters which can deal with multiple binary problems and Section 4.3.1.4 describes methods which are related with other issues, such as robustness. A summary of all the filters reviewed in this subsection can be found in Table 4.1.

4.3.1.1 Information Theory

Firstly, the methods based on information theory will be presented, which despite being conceived years ago, are still the focus of much attention. A novel filter framework is presented by [40] to select an optimal feature subset based on a maximum weight and minimum redundancy (MWMR) criterion. The weight of each feature indicates its importance for some ad hoc tasks (e.g., clustering or classification) and the redundancy represents the correlation among features. With this method it is possible to select the feature subset in which the features are most beneficial to the subsequent tasks while the redundancy among them is minimal. Experimental results on five datasets (two of them based on DNA microarray) demonstrated the advantage and efficiency of MWMR.

In a previous work [41], a statistical dependence measure is presented for gene selection in the context of DNA microarray classification. The proposed method is also based on a maximum relevance minimum redundancy approach, but it uses a simple measure of monotone dependence ($\mathscr{M}_d$) to quantify both relevance and redundancy. $\mathscr{M}_d$ was compared against the well-known minimum redundancy maximum relevance (mRMR) method, and was shown to obtain performance better than or equal to on binary datasets.

Also related with information theory, [42] introduced MASSIVE, a new information-theoretic filter approach for mining microarray data. This filter relies on a criterion which consists of maximizing a term appearing both in the lower bound and in the upper bound of the mutual information of a subset. The experimental results showed that the proposed method is competitive with five state-of-the-art approaches.

An entropic filtering algorithm (EFA) [43] was proposed as a fast feature selection method based on finding feature subsets that jointly maximize the normalized multivariate conditional entropy with respect to the classification ability of tumors. The solutions achieved are of comparable quality to previous results, obtained in a maximum of half an hour computing time and using a very low number of genes.

Song et al. [44] introduced a framework for feature selection based on dependence maximization between the selected features and the labels of an estimation problem, using the Hilbert-Schmidt Independence Criterion. Their proposed method, BAHSIC, is a filter method that demonstrated good performance on microarray data, compared with more specialized methods.

4.3.1.2 Discretization

After presenting the measures related with information theory, the topic of discretization [45] related to feature selection will be discussed. Although the use of a feature selection method when dealing with microarray data is a common practice, discretization has not received the same amount of attention. In [46] the authors proposed not only new techniques for feature selection, but they also added a previous discretization step. They performed scalar feature discretization with the well-known Linde-Buzo-Gray algorithm, using a stopping criterion based on bit allocation. Then, the feature selection step applies a simple unsupervised criterion with indicators on the original numeric features and on the discretized features. They also devised two efficient relevance/redundancy feature selection algorithms (RFS and RRFS) in order to remove redundant features.

The necessity of a prior discretization of the data is introduced for two main reasons: the first one is to help the filtering process and the second one is related to the large number of genes with very unbalanced values present in microarray datasets [35]. Results on ten datasets demonstrated that the combination method, discretizer+filter, outperformed the results achieved by previous approaches, in some cases with improvements in the classification accuracy and decreases in the number of genes needed.

4.3.1.3 Multiple Binary Problems

The same scheme of discretizer+filter was employed again [47], but in this case to be applied only to multiclass datasets. While studies on feature selection using the multiclass approach (a method that can deal directly with multiple classes) are relatively frequent in the literature [35, 41, 42, 44, 46, 48, 49], very few studies employ the multiple binary subproblems approach. Two well-known methods were employed for generating binary problems from a multiple class dataset: one versus one and one versus rest. The methodology was applied on 21 datasets, including a microarray dataset (Leukemia). On this dataset, the best results were obtained when applying feature selection. Specifically, the one versus rest approach obtained promising accuracy results along with a drastic reduction in the number of features needed.

Student et al. [50] also proposed a method based on Partial Least Squares (PLS) and decomposition to a set of two-class subproblems, again using one versus one and one versus rest. They state that it is more effective to solve a multiclass feature selection by splitting it into a set of two-class problems and merging the results in one gene list. In this way, they obtained a very good accuracy rate and stability, while providing easy interpretation of the results by biologists.

4.3.1.4 Other Approaches

Robustness is a trending issue on feature selection. In [48] a new robust feature selection method (RFS) is proposed emphasizing joint $\ell_{2,1}$-norm minimization on both loss function and regularization. This method is robust to outliers and also efficient in calculation.

Finally, a very interesting and novel filter approach was proposed by [49] based on so-called multi-task learning. When the number of labelled microarrays is particularly small (e.g., less than 10), the amount of available information diminishes to the level that even the most carefully designed classification approaches are bound to outperform. An alternative approach is to utilize information from the external microarray datasets, so accuracy on the target classification task can be significantly increased if data from the auxiliary tasks are consulted during learning. The multi-task filter (M_FS) was evaluated on microarray data showing that this method is successful when applied in conjunction with both single-task and multi-task classifiers.

4.3.2 Wrappers

As mentioned before, the wrapper approach has not received the same amount of attention as the filter methods, due to its high computational cost. As the number of features grows, the space of feature subsets grows exponentially, something that becomes a critical aspect when there are tens of thousands of features. Besides, they have the risk of overfitting due to the small sample size of microarray data. As a result, the wrapper approach has been mostly avoided in the literature.

Some works using the wrapper approach can be found in the earliest years of the investigation of microarray data. Notice that in a typical wrapper, a search is conducted in the space of genes, evaluating the goodness of each found gene subset

Table 4.1: Filter methods used on microarray data. Type of evaluation (ranker/subset) and type of data (binary/multiclass)

Method	Original Ref.	Type (r/s)	Data (b/m)
BAHSIC	[44]	r	m
Discretizer+filter	[35, 47]	s	m
EFA	[43]	s	b
$\mathscr{M}_d$	[41]	r	m
M_FS	[49]	r	m
MASSIVE	[42]	r	m
MWMR	[40]	s	b
PLS	[50]	r	m
RFS	[46]	r	m
RFS	[48]	r	m
RRFS	[46]	r	m

by the estimation of the accuracy percentage of the specific classifier to be used, training the classifier only with the found genes. For example, [51] evaluated classical wrapper search algorithms (sequential forward and backward selection, floating selection and best-first search) on three microarray datasets. Another example has been provided by [16], in which an incremental wrapper called BIRS is presented for gene selection. Although the use of wrappers on microarray data has not evolved in the same way as the other feature selection methods, some examples were found in the last few years.

Sharma et al. [52] proposed an algorithm called successive feature selection (SFS). It is well known that most of the conventional feature selection algorithms (e.g., individual ranking and forward selection schemes) have the drawback that a weakly ranked gene that could perform well in terms of classification with an appropriate subset of genes will be left out of the selection. Trying to overcome this shortcoming, the proposed SFS consists of first partitioning the features into smaller blocks. Once the top features from each of the blocks are obtained according to their classification performance, they are compared among themselves to obtain the best feature subset. This algorithm provides high classification accuracy on several DNA microarray datasets.

An evolutionary wrapper method was presented in [53], called (GA-KDE-Bayes). It uses a nonparametric density estimation method and a Bayesian classifier. The authors state that nonparametric methods are a good alternative for scarce and sparse data, as in bioinformatics problems, since they do not make any assumptions about its structure and all the information comes from data itself. Results on six microarray datasets showed better performance than others presented in the literature.

Table 4.2 presents the wrapper methods described, along with the original reference, the type of evaluation (ranker or subset) and the type of data that they can deal with (binary or multiclass).

Table 4.2: Wrapper methods used on microarray data. Type of evaluation (ranker/subset) and type of data (binary/multiclass)

Method	Original Ref.	Type (r/s)	Data (b/m)
GA-KDE-Bayes	[53]	s	b
SPS	[52]	s	m

4.3.3 Embedded

Despite their lower time consumption, a main disadvantage of the filter approach is the fact that it does not interact with the classifier, usually leading to worse performance results than wrappers. However, it has been seen that the wrapper model comes with a expensive computational cost, especially aggravated by the high dimensionality of microarray data. An intermediate solution for researchers is the use of embedded methods, which use the core of the classifier to establish a criteria to rank features. Probably the most famous embedded method is Support Vector Machine based on Recursive Feature Elimination (SVM-RFE), proposed by Guyon et al. [54] to specifically deal with gene selection for cancer classification. This method soon joined the group of algorithms which represent the state of the art for gene selection, and therefore multiple extensions and modifications of it have been proposed. Next, we will describe several embedded approaches designed to deal with microarray data that we found reviewing the up-to-date literature (for a summary of them, consult Table 4.3).

Maldonado et al. [55] introduced a new embedded method. It simultaneously selects relevant features during classifier construction by penalizing each feature's use in the dual formulation of support vector machines (SVM). The approach is called kernel-penalized SVM (KP-SVM) and it optimizes the shape of an anisotropic RBF Kernel, eliminating features that have low relevance for the classifier. The experiments on two benchmark microarray datasets and two real-world datasets showed that KP-SVM outperformed the alternative approaches and determined consistently fewer relevant features.

In [56] the authors proposed a FOIL (First Order Inductive Learner) rule-based feature subset selection algorithm, called FRFS. This method firstly generates the FOIL classification rules using a modified propositional implementation of the FOIL algorithm. Then, it combines the features that appeared in the antecedents of all rules together, and achieves a candidate feature subset that excludes redundant features and reserves the interactive ones. Lastly, it measures the relevance of the features in the candidate feature subset by their proposed new metric CoverRatio and identifies and removes the irrelevant features.

Shah et al. [57] focused not only on obtaining a small number of genes but also on having verifiable future performance guarantees. They investigated the premise of learning conjunctions of decision stumps and proposed three formulations based on different learning principles, which embed the feature selection as a part of the learning process itself. One of their proposals, Probably Approximately Correct (PAC) Bayes, yields competitive classification performance while at the same time utilizing significantly fewer attributes.

The authors in [58] introduced the iterative perturbation method (IFP), which is an embedded gene selector applied to four microarray datasets. This algorithm uses a backward elimination approach and a criterion to determine which features are the least important, relying on the classification performance impact that each feature has when perturbed by noise. If adding noise leads to a big change in the classification performance, then the feature is considered relevant. The IFP approach resulted in comparable or superior average class accuracy when compared to well-studied SVM-RFE on three out of the four datasets.

To overcome the problem of the imbalance of some microarray datasets, a new embedded method based on the random forest algorithm is presented in [59]. Its strategy is composed of different methods and algorithms. First, an algorithm to find the best training error cost for each class is run, in order to deal with the imbalance of the data. Then, random forest is run to select the relevant features. Finally, a strategy to avoid overfitting is also applied. The method was designed ad hoc to deal with a complex gene expression dataset for Leukemia malignancy, showing a very acceptable outcome.

Table 4.3: Embedded methods used on microarray data. Type of evaluation (ranker/subset) and type of data (binary/multiclass)

Method	Original Ref.	Type (r/s)	Data (b/m)
FRFS	[56]	s	m
IFP	[58]	s	b
KP-SVM	[55]	s	m
PAC-Bayes	[57]	r	b
Random Forest	[59]	s	m

4.3.4 Other Algorithms

Nowadays, the trend is to use not only classical feature selection methods (filters, wrappers and embedded) but also to focus on new combinations such as hybrid or ensemble methods.

Hybrid methods usually combine two or more feature selection algorithms of different conceptual origin in a sequential manner. Two of the most famous feature selection methods for microarray data were combined in [60]: SVM-RFE and mRMR. Their authors propose an approach that incorporates a mutual information-based mRMR filter in SVM-RFE to minimize the redundancy among selected genes.

Their approach improved the accuracy of classification and yielded smaller gene sets compared with mRMR and SVM-RFE, as well as other popular methods.

Shreem et al. [61] also used mRMR in their hybrid method. In this case, the proposed approach combines ReliefF, mRMR and GA (Genetic Algorithm) coded as R-m-GA. In the first stage, the candidate gene set is identified by applying ReliefF. Then, the redundancy is minimized with the help of mRMR, which facilitates the selection of effectual gene subset from the candidate set. In the third stage, GA with classifier (used as a fitness function by the GA) is applied to select the most discriminating genes. The proposed method is capable of finding the smallest gene subset that offers the highest classification accuracy.

Chuang et al. [62] proposed a hybrid method called CFS-TGA, which combines correlation-based feature selection (CFS) and the Taguchi-genetic algorithm, where the k-nearest neighbor served as a classifier. The proposed method obtained the highest classification accuracy in ten out the 11 gene expression datasets it was tested on.

Lee et al. [63] developed another hybrid method. It first uses a genetic algorithm with dynamic parameter setting (GADP) to generate a number of subsets of genes and to rank the genes according to their occurrence frequencies in the gene subsets. Then, χ^2 is used to select a proper number of the top-ranked genes for data analysis. Finally, an SVM is employed to verify the efficiency of the selected-genes. The experimental results on six microarray datasets showed that the GADP method is better than the existing methods in terms of the number of selected genes and the prediction accuracy.

Leung et al. [64] proposed a multiple-filter-multiple-wrapper (MFMW) method. The rationale behind this proposal is that filters are fast but their predictions are inaccurate whilst wrappers maximize classification accuracy at the expense of a formidable computation burden. MFMW is based on previous hybrid approaches that maximize the classification accuracy for a chosen classifier with respect to a filtered set of genes. The drawback of the previous hybrid methods which combine filters and wrappers is that the classification accuracy is dependent on the choice of specific filter and wrapper. MFMW overcomes this problem by making use of multiple filters and multiple wrappers to improve the accuracy and robustness of the classification.

Ensemble feature selection builds on the assumption that combining the output of multiple experts is better than the output of any single expert. Typically, ensemble learning has been applied to classification, but it has recently been applied to microarray gene selection. An ensemble of filters (EF) is proposed [65]. The rationale of this approach is the variability of results of each available filter over different microarray datasets. That is, a filter can obtain excellent classification results in a given dataset while performing poorly in another dataset, even in the same domain. This ensemble obtains a classification prediction for every different filter conforming the ensemble, and then combines these predictions by simple voting. Experiments on 10 microarray datasets showed that the ensemble obtained the lowest average of classification error for the four classifiers tested. Recently, the same authors introduced new ensembles to improve performance based on different ways of integrating results [36].

From the perspective of pattern analysis, researchers must focus not only on classification accuracy but also on producing a stable or robust solution. Trying to improve the robustness of feature selection algorithms, Yang et al. [66] proposed an ensemble method called multicriterion fusion-based recursive feature elimination (MCF-RFE). Experimental studies on microarray datasets demonstrated that the MCF-RFE method outperformed the commonly used benchmark feature selection algorithm SVM-RFE both in classification performance and stability of feature selection results.

Abeel et al. [67] are also concerned with the analysis of the robustness of biomarker selection techniques. For this, they proposed a general experimental setup for stability analysis that can be easily included in any biomarker identification pipeline. In addition, they also presented a set of ensemble feature selection methods improving biomarker stability and classification performance on four microarray datasets. They used recursive feature elimination (RFE) as a baseline method and a bootstrapping method to generate diversity in the selection. Then, two different schemes were proposed to aggregate the different rankings of features. Their findings were that when decreasing the number of selected features, the stability of RFE tends to degrade while ensemble methods offer significantly better stability.

Ye et al. [68] proposed a stratified sampling method to select the feature subspaces for random forest (SRF). The key idea is to stratify features into two groups. One group will contain strong informative features and the other weak informative features. Then, for feature subset selection, features are randomly chosen from each group proportionally. The advantage of stratified sampling is that it can ensure that each subspace contains enough informative features for classification in high-dimensional data.

Clustering methods for microarray data have been also recently proposed. Most of the gene selection techniques are based on the assumption of the independence between genes (actually a typical approach is to rank them individually). However, it is well known that genes interact with each other through gene regulative networks. To overcome this problem, Lovato et al. [69] presented a novel feature selection scheme, based on the Counting Grid (GC) model, which can measure and consider the relation and the influence between genes.

Song et al. [70] presented a fast clustering-based feature selection algorithm (FAST) which works in two steps. In the first step, features are divided into clusters by using graph-theoretic clustering methods. In the second step, the most representative feature that is strongly related to target classes is selected from each cluster to form a subset of features. Since features in different clusters are relatively independent, the clustering-based strategy of FAST has a high probability of producing a subset of useful and independent features. The exhaustive evaluation carried out on 35 real-world datasets (14 of them in the microarray domain) demonstrated that FAST not only produced smaller subsets of features, but also improved the performances for four types of classifiers.

Table 4.4: Other feature selection methods used on microarray data. Type of evaluation (ranker/subset) and type of data (binary/multiclass)

Method	Original Ref.	Type (r/s)	Data (b/m)
CFS-TGA	[62]	s	m
E1-cp	[36]	s	b
E1-nk	[36]	s	b
E1-ns	[36]	s	b
E2	[36]	s	b
Ensemble RFE	[67]	s	b
EF	[65]	s	m
FAST	[70]	s	m
GADP	[63]	s	m
GC	[69]	r	b
MCF-RFE	[66]	s	b
MFMW	[64]	s	b
R-m-GA	[61]	s	b
SRF	[68]	s	m
SVM-RFE with MRMR	[60]	r	b

Table 4.4 depicts a summary of the methods presented in this section. The original reference is displayed, as well as the type of evaluation (ranker or subset) and the type of the data they can deal with (binary or multiclass).

4.4 A Framework for Feature Selection Evaluation in Microarray Datasets

The goal of performing feature selection on microarray data can be twofold: class prediction or biomarkers identification. If the goal is class prediction, there is a demand for machine learning techniques such as supervised classification. However, if the objective is to find informative genes, the classification performance is ignored and the selected genes have to be individually evaluated. The experiments that will be presented in this section are focused on class prediction, which is an important reason to use feature selection methods in microarray analysis. The typical microarray pipeline is formed by a feature selection step, followed by a classification stage and providing an error estimation, as seen in Figure 4.3.

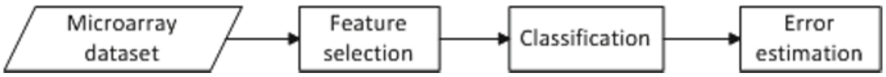


Fig. 4.3: DNA microarray classification pipeline

The rest of this section is devoted to the importance of the validation techniques usually applied on microarray data and to analyze the characteristics of the datasets whilst providing classification accuracy results obtained with classical feature selection methods.

4.4.1 Validation Techniques

To evaluate the goodness of the selected set of genes, it is necessary to have an independent test set with data which have not been seen by the feature selection method. In some cases, the data come originally distributed into training and test sets, so the training set is usually employed to perform the feature selection process and the test set is used to evaluate the appropriateness of the selection. However, not all the datasets found in the literature are originally partitioned. For overcoming this issue, several validation techniques exist, where the most used in the microarray domain are k-fold cross-validation, leave-one-out cross-validation, bootstrap and holdout validation (see Appendix A, Section A.3).

Probably cross-validation would be the most famous technique. However, it has been shown [5] that it can potentially introduce dataset shift, a harmful factor that is often not taken into account and can result in inaccurate performance estimation. To solve this problem, *Distribution optimally balanced stratified cross-validation* (DOB-SCV) [5] is based on the idea that by assigning close-by examples to different folds, each fold will end up with enough representatives of every region, thus avoiding dataset shift. To achieve this goal, DOB-SCV starts on a random unassigned example, and then finds its $k-1$ nearest unassigned neighbors of the same class. Once it has found them, it assigns each of those examples to a different fold. The process is repeated until all examples are assigned.

The selection of a validation technique on the microarray domain is not an easy-to-solve question. This is due to the fact that microarray data is characterized by an extremely large number of features and comparatively small number of samples. This situation is commonly referred to as a *small-sample* scenario, which means that application of traditional pattern recognition methods must be carried out with good judgment to avoid pitfalls [18]. A key point for microarray classification is that error estimation is greatly impacted by small samples [17], so the choice of a validation technique must be further discussed. In fact, there are several works in the literature dealing with this issue. Ambroise et al. [71] recommended the use of 10-fold cross validation rather than leave-one-out, and, concerning the bootstrap, they suggest using the so-called 0.632+ bootstrap error estimate (which weighs the bootstrapped resubstitution error), designed to handle overfitted prediction rules. Braga et al. [72] performed an extensive simulation study by comparing cross-validation, resubstitution and bootstrap estimation. They stated that while cross-validation error estimation is much less biased than resubstitution, it displays excessive variance, which makes individual estimates unreliable for small samples. Bootstrap methods provide improved performance relative to variance, but at a high computational cost and often with increased bias.

In this situation, the so-called best validation technique for microarray data does not exist. In fact, reviewing the recent literature one can find examples of the four methods described above. *k*-fold cross-validation is a common choice [42, 44, 48, 57, 58, 61, 68], as is holdout validation [43, 46, 52, 55, 59, 63, 69]. Bootstrap sampling was used less [50, 60, 66], probably due to its high computational cost, and there are also some representatives of leave-one-out cross-validation [62, 64].

4.4.2 On the Datasets Characteristics

In this study, nine widely used binary microarray datasets have been considered, which are available for download in [73, 74, 75]. The reason for choosing binary datasets is because they are much more common in the literature than the multiclass ones. As a matter of fact, a typical microarray dataset consists of distinguishing between having a given cancer or not; therefore the great majority of the datasets are binary. Tables 4.5 and 4.6 show the imbalance ratio (IR) and the F1 (*maximum Fisher's discriminant ratio*, see Chapter 1) of the datasets used, whereas more details about them (such as number of features or samples) can be found in Appendix A, Section A.2.3. The imbalance ratio [76] is defined as the number of negative class examples that are divided by the number of positive class examples, where a high level indicates that the dataset is highly imbalanced. F1 (see Section 4.2.3) checks for overlapping among the classes where the higher the F1 is, the more separable the data is. Notice that for the datasets in Table 4.5, which come originally divided into training and test sets, these measures are shown for both partitions. For these two datasets, it is shown that both of them present more imbalance in the test set than in the training set, especially in the case of Prostate dataset. As for the F1 measure, the higher amount of overlapping occurs on the Breast training set, with a value of 0.68. Regarding the information depicted in Table 4.6, the most unbalanced dataset is GLI and the one more affected by overlapping is SMK, with a F1 value of 0.41.

Table 4.5: Imbalance ratio and F1 of the binary datasets used in the holdout experimental study

	Train		Test	
Dataset	IR	F1	IR	F1
Breast	1.29	0.68	1.71	4.98
Prostate	1.04	2.05	2.78	11.35

Table 4.6: Imbalance ratio and F1 of the binary datasets used in the *k*-fold cross-validation experimental study

Dataset	IR	F1
Brain	2.00	0.89
CNS	1.86	0.45
Colon	1.82	1.08
DLBCL	1.04	2.91
GLI	2.27	2.35
Ovarian	1.78	6.94
SMK	1.08	0.41

4.4.3 Feature Selection Methods

Seven classical feature selection methods were chosen to be applied on this study: CFS, FCBF, INTERACT, Information Gain, ReliefF, mRMR and SVM-RFE (see Chapter 2). All of them are available in the well-known Weka tool [77], except for mRMR filter, whose implementation is available for Matlab. These methods are used extensively in the literature, and their performance can serve as a reference for the interested reader.

4.4.4 Evaluation Measures

In order to evaluate the behavior of the feature selection methods after applying a classifier, three well-known measures are used: accuracy, sensitivity and specificity (see Appendix A, Section A.7). In general, sensitivity indicates how well the test predicts the actual positives (e.g., the percentage of cancer patients who are correctly identified as having the condition) while specificity measures how well the test identifies the negatives (e.g., the percentage of healthy patients who are correctly identified as not having cancer). A perfect predictor would be described as 100% sensitive (e.g., predicting all patients with cancer as such) and 100% specific (e.g., not predicting any patient from the healthy group as having cancer). Accuracy is expected to measure how well the test predicts both categories.

4.5 A Practical Evaluation: Analysis of Results

The goal of this section is to perform an experimental study using the most representative binary datasets, described in Section 4.4.2 and some classical feature selection methods. To evaluate the adequacy of these methods over microarray data, three well-known classifiers were chosen: C4.5, naive Bayes and SVM (see Appendix A). As reported in Section 4.4.1, there is no consensus in the literature about which validation technique to use when dealing with microarray data. In light of these facts, two studies will be performed. In the first one, a holdout validation will be applied on those datasets which come originally divided in training and test datasets. As revealed in Section 4.2.4, the training and test data of those datasets were extracted under different conditions, which means an added challenge for the machine learning methods. If the two sets are joined in an unique dataset (e.g., for later applying a *k*-fold cross-validation), the new situation would be easier for the learner, and this particular characteristic of microarray data would be overlooked. The second study will consist of applying a five-fold cross-validation over those datasets which provide a unique training test, where five folds have been chosen because with the standard value of 10, for some datasets the test set would remain with only a couple of samples. However, as mentioned in Section 4.4.1, in some cases cross-validation can potentially introduce dataset shift; so we will include DOB-SCV in the experimental study to try to overcome this problem.

The first three feature selection methods (CFS, FCBF and INTERACT) return a subset of features, whilst the remaining four (IG, ReliefF, mRMR and SVM-RFE) provide an ordered ranker of the features. For the ranker methods, the performance when retaining the top 10 and 50 features is shown. For those methods which return a subset of features, the number of features selected for each dataset is revealed in Table 4.7. Notice that for the datasets involved in the cross-validation study (Brain, CNS, Colon, DLBCL, Gli85, Ovarian and Smk), this number is the mean average of the number of features selected in each fold. Since two types of partitions are tried, both values are shown in the table (regular cross-validation / DOB-SCV).

Table 4.7: Number of features selected by subset methods on binary datasets

Method	Brain	Breast	CNS	Colon	DLBCL	Gli85	Ovarian	Prostate	Smk
CFS	36/49	130	44/44	24/25	61/65	141/156	35/33	89	107/103
FCBF	1/1	99	33/35	14/15	35/37	116/118	27/26	77	50/55
INT	36/49	102	33/34	14/16	45/51	117/123	32/31	73	51/51

4.5.1 Holdout Validation Study

This section reports the experimental results achieved over the binary datasets that are originally divided into training and test sets (see Table 4.5). Tables 4.8, 4.9 and 4.10 show the results achieved by C4.5, naive Bayes and SVM, respectively. These tables depict the classification accuracy (Ac), sensitivity (Se) and specificity (Sp) on the test datasets. For the sake of comparison, the first row shows those values without applying feature selection techniques. Notice that the C4.5 algorithm does a feature selection because not all the attributes are considered when constructing the tree. The best results for dataset and classifier are marked in boldface.

Analyzing these tables, it is easy to note that the results obtained with SVM outperformed notably those achieved by C4.5 or naive Bayes. In fact, there has been in the literature a clear tendency to use SVM as the standard de facto method to estimate performance measures, and the authors in [39] stated that SVM seems to be undoubtedly superior to several other classifiers. As mentioned in Section 4.2.4, the Prostate dataset suffers from dataset shift, since the test dataset was extracted from a different experiment, and apparently C4.5 and naive Bayes classifiers cannot solve this problem and opted to assign all the examples to the majority class.

Table 4.8: Experimental results for the C4.5 classifier on binary datasets after performing holdout validation

		Breast			Prostate		
		Ac	Se	Sp	Ac	Se	Sp
no FS		0.74	**1.00**	0.58	0.26	**1.00**	0.00
CFS		0.68	0.71	0.66	0.26	**1.00**	0.00
FCBF		0.58	0.28	0.75	0.26	**1.00**	0.00
INT		**0.79**	0.71	0.83	0.26	**1.00**	0.00
IG	#10	0.47	0.28	0.58	0.26	**1.00**	0.00
	#50	0.53	0.42	0.58	0.29	**1.00**	0.04
ReliefF	#10	0.58	0.28	0.75	0.26	**1.00**	0.00
	#50	0.42	0.71	0.25	0.29	**1.00**	0.04
SVM-RFE	#10	0.58	**1.00**	0.33	0.32	**1.00**	0.08
	#50	0.58	**1.00**	0.33	0.26	**1.00**	0.00
mRMR	#10	0.58	0.71	0.50	**0.41**	0.88	**0.24**
	#50	0.74	0.42	**0.91**	0.35	**1.00**	0.12

Table 4.9: Experimental results for the naive Bayes classifier on binary datasets after performing holdout validation

		Breast			Prostate		
		Ac	Se	Sp	Ac	Se	Sp
no FS		0.37	**1.00**	0.00	**0.26**	**1.00**	0.00
CFS		0.37	**1.00**	0.00	**0.26**	**1.00**	0.00
FCBF		0.37	**1.00**	0.00	**0.26**	**1.00**	0.00
INT		0.37	**1.00**	0.00	**0.26**	**1.00**	0.00
IG	#10	0.32	0.85	0.00	**0.26**	0.88	0.04
	#50	0.37	**1.00**	0.00	0.24	0.88	0.00
ReliefF	#10	0.74	0.71	0.75	0.21	0.55	**0.08**
	#50	**0.89**	0.85	**0.91**	0.21	0.77	0.00
SVM-RFE	#10	0.68	0.85	0.58	0.18	0.55	0.04
	#50	0.63	**1.00**	0.41	**0.26**	**1.00**	0.00
mRMR	#10	0.37	**1.00**	0.00	**0.26**	**1.00**	0.00
	#50	0.37	**1.00**	0.00	**0.26**	**1.00**	0.00

Table 4.10: Experimental results for the SVM classifier on binary datasets after performing holdout validation

		Breast			Prostate		
		Ac	Se	Sp	Ac	Se	Sp
no FS		0.58	0.85	0.41	0.53	**1.00**	0.36
CFS		0.68	0.85	0.58	**0.97**	**1.00**	**0.96**
FCBF		0.58	0.28	0.75	**0.97**	**1.00**	**0.96**
INT		0.74	0.71	0.75	0.71	**1.00**	0.60
IG	#10	0.58	0.71	0.50	**0.97**	**1.00**	**0.96**
	#50	0.79	0.57	**0.91**	**0.97**	**1.00**	**0.96**
ReliefF	#10	**0.84**	**1.00**	0.75	0.94	0.88	**0.96**
	#50	**0.84**	0.85	0.83	**0.97**	**1.00**	**0.96**
SVM-RFE	#10	0.68	**1.00**	0.50	0.79	**1.00**	0.72
	#50	0.68	**1.00**	0.50	0.74	**1.00**	0.64
mRMR	#10	0.63	0.71	0.58	0.44	**1.00**	0.24
	#50	0.68	0.71	0.66	0.91	0.77	**0.96**

4.5.2 Cross-validation Study

This section reveals the classification results obtained when applying the well-known cross-validation technique. A five-fold cross-validation was performed over the binary datasets presented in Table 4.6, which have only the training set available. Since in some cases cross-validation can potentially introduce the dataset shift problem, another strategy has been used. Distribution optimally balanced stratified cross-validation (DOB-SCV) tries to avoid dataset shift by assigning close-by examples to different folds. Section 4.5.2.1 analyzes the behavior of the feature selection methods studied on the datasets, whilst Section 4.5.2.2 compares the performance of regular cross-validation against DOB-SCV. Finally, Section 4.5.2.3 will analyze the influence of the datasets' characteristics.

Tables 4.11–4.16 show the results achieved by C4.5, naive Bayes and SVM, for the two types of cross-validation. The results shown in the tables are the average test results for the five folds, depicting the classification accuracy (Ac), sensitivity (Se) and specificity (Sp). Again, the first row reports those values without applying feature selection and the best results are marked in boldface.

4.5.2.1 Analysis of Algorithms

This subsection aims to analyze the behavior of the feature selection methods as well as the influence of the classifier on the studied datasets. Some interesting conclusions can be extracted by looking at the results reported by Tables 4.11–4.16.

1. The best performances are obtained by SVM and naive Bayes classifiers. As mentioned above, some studies [39] stated the superiority of SVMs over other classifiers. On the other hand, the performance of C4.5 may be affected by its embedded feature selection, in some cases leading to an extremely reduced set of features which can degrade the classification accuracy.
2. Focusing on the feature selection methods, on average for all datasets, the subset filters show outstanding behavior, especially CFS and INTERACT. It is surprising that SVM-RFE did not achieve the best results when combined with the SVM classifier, but the poor performance of the ranker methods can be explained by the restriction of having to establish a threshold for the number of features to retain. In the case of the subsets filters, the number of features which conform to the final subset of features is the optimal one for a given dataset and method. However, in the case of rankers, since this number has to be set a priori, it may prove too small or too large, the main disadvantage of using these types of methods.

Table 4.11: Experimental results for the C4.5 classifier on binary datasets after performing regular five-fold cross-validation

			Brain	CNS	Colon	DLBCL	Gli85	Ovarian	Smk	Avg
No FS		Ac	**1.00**	0.58	0.74	0.70	0.75	0.97	0.65	0.77
		Se	**1.00**	0.64	0.60	0.69	0.81	0.95	0.66	0.77
		Sp	**1.00**	0.48	0.82	0.70	0.63	0.98	0.62	0.75
CFS		Ac	**1.00**	0.62	0.79	0.75	0.79	0.98	0.64	**0.79**
		Se	**1.00**	0.64	0.68	0.78	0.81	0.95	0.56	0.78
		Sp	**1.00**	**0.58**	0.85	0.71	0.75	**0.99**	**0.71**	**0.80**
FCBF		Ac	0.86	0.48	0.79	0.73	0.82	**0.99**	0.61	0.75
		Se	0.80	0.49	0.64	0.74	0.86	**0.99**	0.65	0.74
		Sp	0.86	0.50	0.87	0.70	0.75	**0.99**	0.56	0.75
INT		Ac	**1.00**	0.55	0.79	0.70	0.78	0.98	0.59	0.77
		Se	**1.00**	0.54	0.72	0.74	0.81	0.98	0.51	0.76
		Sp	**1.00**	**0.58**	0.82	0.66	0.71	0.98	0.66	0.77
IG	#10	Ac	0.71	0.62	0.72	0.75	**0.85**	0.96	0.60	0.74
		Se	0.70	0.69	0.78	0.79	0.88	0.93	0.71	0.78
		Sp	0.70	0.48	0.70	0.71	**0.79**	0.97	0.48	0.69
	#50	Ac	0.81	0.63	**0.84**	0.73	0.81	0.96	0.65	0.78
		Se	0.70	0.67	**0.83**	0.69	0.86	0.96	0.62	0.76
		Sp	0.87	**0.58**	0.85	0.74	0.71	0.97	0.67	0.77
ReliefF	#10	Ac	0.72	0.47	0.72	**0.85**	**0.85**	0.97	0.65	0.75
		Se	0.20	0.59	0.50	0.83	0.88	0.94	**0.80**	0.68
		Sp	**1.00**	0.25	0.85	**0.87**	0.77	**0.99**	0.47	0.74
	#50	Ac	0.62	0.53	0.82	0.73	0.82	**0.99**	0.61	0.73
		Se	0.20	0.60	0.68	0.74	0.88	**0.99**	0.61	0.67
		Sp	0.86	0.44	**0.90**	0.70	0.70	**0.99**	0.62	0.74
SVM-RFE	#10	Ac	0.57	**0.65**	0.71	0.81	0.81	0.98	0.60	0.73
		Se	0.00	**0.74**	0.60	0.82	0.85	0.98	0.65	0.66
		Sp	0.87	0.48	0.77	0.79	0.75	0.98	0.55	0.74
	#50	Ac	0.70	0.57	0.80	0.82	0.79	0.98	0.65	0.76
		Se	**1.00**	0.61	0.77	**0.84**	0.83	**0.99**	0.62	**0.81**
		Sp	0.56	0.49	0.82	0.79	0.70	0.98	0.66	0.72
mRMR	#10	Ac	0.86	0.55	0.82	0.75	0.79	0.98	**0.68**	0.77
		Se	0.90	0.72	0.68	0.79	0.86	0.96	0.71	0.80
		Sp	0.87	0.23	**0.90**	0.70	0.61	**0.99**	0.64	0.70
	#50	Ac	0.86	0.58	0.82	0.73	0.80	0.97	0.62	0.77
		Se	0.90	0.70	0.77	0.69	**0.91**	0.96	0.66	0.80
		Sp	0.87	0.39	0.85	0.74	0.54	0.98	0.57	0.71

4.5.2.2 Cross-validation vs. DOB-SCV

The purpose of this subsection is to check the adequacy of performing DOB-SCV instead of regular five-fold cross-validation. On average for all datasets, DOB-SCV obtains better results than regular cross-validation for SVM and naive Bayes classifiers, which are the classifiers which showed the best overall performance. It is interesting to focus on the case of the Brain dataset, which represents a high imbalance and an important amount of overlapping, as can be seen in Table 4.6. For this dataset, DOB-SCV outperforms regular cross-validation for SVM and naive Bayes classifiers, although the highest accuracy was achieved by C4.5 combined with the regular cross-validation. In the case of CNS, another complex dataset, there are no important differences between the two validation methods. On the other hand, there is the DLBCL dataset, which is in theory a simpler dataset than CNS and Brain (see

Table 4.12: Experimental results for the C4.5 classifier on binary datasets after performing DOB-SCV with five folds

			Brain	CNS	Colon	DLBCL	Gli85	Ovarian	Smk	Avg
No FS		Ac	**0.92**	0.52	0.72	0.72	0.77	0.96	0.56	0.74
		Se	0.80	0.58	0.55	0.70	0.79	0.93	0.63	0.71
		Sp	**1.00**	0.39	0.80	0.74	0.74	0.97	0.48	0.73
CFS		Ac	**0.92**	0.52	0.79	0.80	0.75	0.96	0.59	0.76
		Se	0.80	0.53	0.65	**0.82**	0.77	0.93	0.59	0.73
		Sp	**1.00**	0.48	0.87	0.78	0.70	**0.98**	0.59	0.77
FCBF		Ac	0.72	0.52	0.81	0.72	0.76	**0.98**	0.59	0.73
		Se	0.70	0.56	0.69	0.70	0.79	**0.98**	0.53	0.71
		Sp	0.76	0.44	0.87	0.74	0.70	**0.98**	0.65	0.74
INT		Ac	**0.92**	0.53	0.81	0.74	0.71	0.97	0.67	0.76
		Se	0.80	0.51	0.69	0.65	0.69	0.94	0.58	0.69
		Sp	**1.00**	**0.56**	0.87	0.84	0.74	**0.98**	**0.75**	**0.82**
IG	#10	Ac	0.72	0.60	0.79	0.78	0.82	0.96	0.62	0.76
		Se	0.80	0.67	0.59	0.74	0.81	0.94	0.61	0.74
		Sp	0.73	0.49	**0.90**	0.83	**0.85**	0.96	0.63	0.77
	#50	Ac	0.77	0.54	**0.84**	0.80	0.81	**0.98**	0.60	0.76
		Se	0.80	0.59	**0.74**	**0.82**	0.79	**0.98**	0.58	0.76
		Sp	0.80	0.44	**0.90**	0.79	**0.85**	0.97	0.62	0.77
ReliefF	#10	Ac	0.48	0.55	0.77	**0.84**	0.81	0.96	0.66	0.73
		Se	0.10	**0.76**	0.69	**0.82**	**0.91**	0.93	0.67	0.70
		Sp	0.63	0.15	0.82	0.87	0.57	0.98	0.64	0.67
	#50	Ac	0.53	0.56	0.76	0.80	**0.85**	**0.98**	0.68	0.74
		Se	0.60	0.69	0.63	**0.82**	0.86	**0.98**	**0.75**	0.76
		Sp	0.50	0.34	0.82	0.79	0.81	**0.98**	0.60	0.69
SVM-RFE	#10	Ac	0.59	0.56	0.76	0.82	0.78	**0.98**	0.60	0.73
		Se	0.40	0.66	0.64	0.73	0.79	0.96	0.66	0.69
		Sp	0.70	0.37	0.82	**0.92**	0.74	**0.98**	0.53	0.72
	#50	Ac	0.70	**0.62**	0.78	0.78	0.76	0.97	0.65	0.75
		Se	0.70	0.65	0.56	0.69	0.76	0.95	0.69	0.71
		Sp	0.73	**0.56**	**0.90**	0.88	0.73	**0.98**	0.61	0.77
mRMR	#10	Ac	0.80	0.60	0.79	0.78	0.80	**0.98**	0.68	**0.78**
		Se	0.80	0.71	0.65	0.78	0.79	**0.98**	**0.75**	**0.78**
		Sp	0.83	0.38	0.87	0.80	0.81	**0.98**	0.61	0.75
	#50	Ac	0.84	0.60	0.82	0.76	0.78	**0.98**	**0.71**	**0.78**
		Se	**0.90**	0.66	0.69	0.78	0.76	**0.98**	0.71	**0.78**
		Sp	0.83	0.47	**0.90**	0.74	0.82	**0.98**	0.70	0.78

Table 4.6). In fact, the accuracy obtained by the classifiers is in most of the cases around 90%. Nevertheless, it is interesting to note that for this dataset, DOB-SCV outperforms, on average, the regular cross-validation. This dataset will be studied in detail in Section 4.5.2.3.

4.5.2.3 Analysis of Datasets Characteristics

As mentioned in Section 4.2, microarray datasets present several problematics such as the imbalance of the data, the overlapping between classes or the dataset shift. Tables 4.5 and 4.6 reported the imbalance ratio and F1 of the datasets studied in this section, which measure the imbalance of the data and the overlapping, respectively. GLI is the most unbalanced dataset, and its highest accuracy was obtained

Table 4.13: Experimental results for the naive Bayes classifier on binary datasets after performing regular five-fold cross-validation

			Brain	CNS	Colon	DLBCL	Gli85	Ovarian	Smk	Avg
No FS		Ac	0.67	0.60	0.55	0.92	0.84	0.93	0.63	0.73
		Se	0.00	0.64	0.69	**0.96**	0.88	0.99	0.60	0.68
		Sp	**1.00**	0.52	0.47	0.88	0.73	0.89	0.66	0.74
CFS		Ac	0.81	0.67	**0.85**	0.90	0.82	**1.00**	0.65	**0.81**
		Se	0.50	0.75	0.76	**0.96**	**0.90**	0.99	0.67	0.79
		Sp	**1.00**	0.54	**0.90**	0.84	0.67	**1.00**	0.62	0.79
FCBF		Ac	0.61	**0.70**	0.80	0.90	0.85	0.99	0.69	0.79
		Se	**1.00**	0.77	0.76	**0.96**	**0.90**	**1.00**	0.72	**0.87**
		Sp	0.40	0.58	0.82	0.84	0.74	0.99	0.65	0.72
INT		Ac	0.81	**0.70**	0.77	0.90	0.82	**1.00**	0.64	**0.81**
		Se	0.50	0.77	0.76	**0.96**	0.88	**1.00**	0.72	0.80
		Sp	**1.00**	0.58	0.77	0.83	0.71	0.99	0.55	0.78
IG	#10	Ac	**0.86**	0.63	0.79	**0.94**	0.85	0.96	0.61	**0.81**
		Se	0.70	0.67	0.72	**0.96**	0.88	0.95	0.59	0.78
		Sp	0.93	0.58	0.82	**0.92**	0.77	0.96	0.64	0.80
	#50	Ac	0.81	0.63	0.77	0.92	0.85	0.98	0.66	0.80
		Se	0.50	0.75	0.76	**0.96**	0.86	0.96	0.67	0.78
		Sp	**1.00**	0.42	0.77	0.88	0.81	0.98	0.65	0.79
ReliefF	#10	Ac	0.20	0.63	0.82	**0.94**	0.86	0.96	0.67	0.73
		Se	0.20	0.72	0.72	**0.96**	0.88	0.95	0.71	0.73
		Sp	0.20	0.48	0.87	**0.92**	0.81	0.96	0.63	0.70
	#50	Ac	0.19	0.67	0.84	0.92	**0.89**	0.98	0.67	0.74
		Se	0.50	0.72	0.77	**0.96**	0.86	0.95	0.72	0.78
		Sp	0.07	0.58	0.87	0.88	**0.97**	0.99	0.61	0.71
SVM-RFE	#10	Ac	0.62	0.68	0.73	0.92	0.82	0.99	**0.71**	0.78
		Se	0.30	0.77	0.61	0.91	0.83	**1.00**	**0.77**	0.74
		Sp	0.76	0.54	0.80	**0.92**	0.81	0.98	0.64	0.78
	#50	Ac	0.67	**0.70**	0.76	0.92	0.88	0.99	0.70	0.80
		Se	0.20	**0.82**	0.69	0.91	0.86	**1.00**	0.73	0.75
		Sp	0.90	0.49	0.80	**0.92**	0.93	0.98	0.65	**0.81**
mRMR	#10	Ac	0.73	0.63	0.80	0.92	0.85	0.99	0.67	0.80
		Se	0.60	0.79	0.78	**0.96**	0.88	0.96	0.68	0.81
		Sp	0.86	0.33	0.82	0.88	0.77	**1.00**	0.65	0.76
	#50	Ac	0.63	0.62	0.80	**0.94**	0.80	0.99	0.67	0.78
		Se	0.20	0.75	**0.86**	**0.96**	0.81	0.98	0.67	0.75
		Sp	0.86	0.38	0.77	**0.92**	0.77	0.99	**0.67**	0.77

by SVM with a regular five-fold cross validation and no feature selection (92%), although the Information Gain filter achieves 91% accuracy and this degradation can be equivalent to missclassifying only one or two samples.

SMK is the dataset which presents a higher level of overlapping between classes, and its maximum classification accuracy is very poor, around 70%. A similar case happens with the CNS dataset, which has also a low value of F1 (see Table 4.6).

Regarding the dataset shift problem and the adequacy of DOB-SCV to solve it, we will analyze in detail the case of the DLBCL dataset. Figure 4.4 displays the 2-D representation of the first two features selected by mRMR in the first fold of a five-fold cross-validation and a 5DOB-SCV for both train and test sets, where different colors stand for different classes. As can be seen, cross-validation can indeed introduce dataset shift. The two first features selected by mRMR obtain a linearly separable problem (see Figure 4.4a) in the train set, but these features are

Table 4.14: Experimental results for the naive Bayes classifier on binary datasets after performing DOB-SCV with five folds

			Brain	CNS	Colon	DLBCL	Gli85	Ovarian	Smk	Avg
No FS		Ac	0.67	0.63	0.58	**0.98**	0.87	0.94	0.63	0.76
		Se	0.00	0.69	0.72	**1.00**	0.91	0.98	0.58	0.70
		Sp	**1.00**	0.54	0.50	0.96	0.77	0.91	0.67	0.77
CFS		Ac	**0.92**	0.67	0.82	0.93	0.85	0.98	0.68	**0.84**
		Se	**0.80**	0.76	0.77	**1.00**	**0.93**	0.98	0.74	0.85
		Sp	**1.00**	0.50	0.85	0.87	0.66	0.98	0.62	0.78
FCBF		Ac	0.71	0.58	0.79	0.96	**0.88**	0.99	0.66	0.80
		Se	**0.80**	0.69	0.77	**1.00**	**0.93**	0.99	0.70	0.84
		Sp	0.70	0.40	0.80	0.92	0.77	**0.99**	0.62	0.74
INT		Ac	**0.92**	0.62	**0.84**	0.93	0.86	0.99	0.70	**0.84**
		Se	**0.80**	0.69	**0.86**	**1.00**	0.91	**1.00**	0.74	**0.86**
		Sp	**1.00**	0.50	0.82	0.87	0.73	**0.99**	0.65	0.79
IG	#10	Ac	**0.92**	0.60	0.77	0.91	0.87	0.96	0.68	0.82
		Se	**0.80**	0.59	0.72	0.96	0.89	0.95	0.66	0.80
		Sp	**1.00**	**0.64**	0.80	0.87	0.81	0.96	**0.71**	**0.83**
	#50	Ac	**0.92**	0.65	0.79	0.96	0.87	0.98	0.70	**0.84**
		Se	**0.80**	0.72	0.82	0.96	0.90	0.96	0.68	0.83
		Sp	**1.00**	0.54	0.77	0.96	0.81	**0.99**	**0.71**	**0.83**
ReliefF	#10	Ac	0.26	0.65	**0.84**	0.93	0.84	0.96	0.67	0.74
		Se	0.30	0.69	0.72	0.96	0.88	0.95	0.71	0.74
		Sp	0.20	0.57	**0.90**	0.91	0.74	0.96	0.62	0.70
	#50	Ac	0.21	0.67	**0.84**	0.96	0.86	0.98	0.68	0.74
		Se	0.30	0.71	0.72	0.96	0.86	0.95	**0.77**	0.75
		Sp	0.13	0.58	**0.90**	0.96	**0.85**	**0.99**	0.59	0.71
SVM-RFE	#10	Ac	0.58	**0.72**	0.76	0.94	0.83	**1.00**	0.70	0.79
		Se	0.20	0.76	0.64	0.91	0.86	**1.00**	0.75	0.73
		Sp	0.73	0.63	0.82	0.96	0.74	**0.99**	0.63	0.79
	#50	Ac	0.71	0.69	0.76	0.92	0.87	0.99	0.68	0.80
		Se	0.10	**0.77**	0.69	**1.00**	0.88	**1.00**	0.76	0.74
		Sp	**1.00**	0.54	0.80	0.84	**0.85**	**0.99**	0.59	0.80
mRMR	#10	Ac	0.75	0.68	0.82	**0.98**	0.87	0.99	**0.71**	0.83
		Se	**0.80**	0.74	0.77	**1.00**	**0.93**	0.98	0.74	0.85
		Sp	0.73	0.58	0.85	0.96	0.73	**0.99**	0.66	0.79
	#50	Ac	0.77	0.67	0.79	**0.98**	0.85	0.98	0.67	0.82
		Se	0.40	0.71	0.82	0.96	0.87	0.96	0.70	0.78
		Sp	**1.00**	0.59	0.77	**1.00**	0.77	**0.99**	0.64	0.82

not so informative in the test set (see Figure 4.4b). However, the partitions created by DOB-SCV do not suffer from dataset shift. In Figure 4.4c, the two first features selected by mRMR in the training set make the problem almost linearly separable and this condition is maintained in the test set. In fact, it has been demonstrated in the previous section that DOB-SCV outperformed the regular cross-validation for this dataset.

The results displayed in this experimental study can be seen to be very dependent on the classifier, the feature selection method, and especially the dataset. Although a detailed analysis of the results is out of the scope of this chapter, it is easy to realize that the large number of problematics present in this type of datasets make the classification task very arduous. In such a situation, studying the particularities of each problem carefully is recommended, although it seems that the best results

Table 4.15: Experimental results for the SVM classifier on binary datasets after performing regular five-fold cross-validation

			Brain	CNS	Colon	DLBCL	Gli85	Ovarian	Smk	Avg
No FS		Ac	**0.68**	0.67	0.77	**0.96**	**0.92**	**1.00**	**0.72**	**0.82**
		Se	0.20	0.82	0.60	0.96	**0.98**	**1.00**	0.79	0.77
		Sp	0.93	0.38	0.87	**0.96**	0.78	**1.00**	0.63	**0.79**
CFS		Ac	0.61	0.62	0.81	0.88	0.88	**1.00**	0.64	0.78
		Se	0.60	0.70	0.69	0.86	0.93	**1.00**	0.66	0.78
		Sp	0.66	0.49	0.87	0.88	0.77	**1.00**	0.61	0.76
FCBF		Ac	0.67	0.65	0.84	0.81	0.87	**1.00**	0.71	0.79
		Se	0.00	0.80	0.73	0.82	0.93	**1.00**	0.76	0.72
		Sp	**1.00**	0.39	0.90	0.79	0.73	**1.00**	0.64	0.78
INT		Ac	0.61	0.62	0.81	0.88	0.88	**1.00**	0.66	0.78
		Se	0.60	0.75	0.64	0.91	0.91	**1.00**	0.69	0.79
		Sp	0.66	0.39	0.90	0.83	0.81	**1.00**	0.63	0.75
IG	#10	Ac	0.48	0.63	0.81	0.94	0.91	0.98	0.64	0.77
		Se	0.00	0.82	0.59	0.96	**0.98**	0.96	0.74	0.72
		Sp	0.70	0.30	**0.92**	0.92	0.74	0.99	0.53	0.73
	#50	Ac	0.66	0.67	**0.85**	0.94	0.86	**1.00**	**0.72**	0.81
		Se	**0.80**	0.77	**0.81**	0.96	0.90	**1.00**	0.73	**0.85**
		Sp	0.66	0.48	0.87	0.92	0.77	0.99	0.70	0.77
ReliefF	#10	Ac	0.50	0.68	0.81	0.94	0.87	0.98	0.69	0.78
		Se	0.00	0.87	0.60	0.96	0.96	0.94	**0.82**	0.74
		Sp	0.73	0.34	**0.92**	0.92	0.66	0.99	0.54	0.73
	#50	Ac	0.35	**0.73**	**0.85**	0.92	0.89	**1.00**	0.69	0.78
		Se	0.00	0.82	0.72	**1.00**	0.93	**1.00**	0.74	0.74
		Sp	0.53	**0.58**	0.92	0.84	0.82	**1.00**	0.64	0.76
SVM-RFE	#10	Ac	0.62	**0.73**	0.73	0.87	0.86	**1.00**	0.70	0.79
		Se	0.20	0.84	0.56	0.87	0.88	**1.00**	0.78	0.73
		Sp	0.86	0.53	0.82	0.88	0.81	**1.00**	0.61	**0.79**
	#50	Ac	0.48	0.72	0.71	0.88	0.89	**1.00**	**0.72**	0.77
		Se	0.20	0.82	0.57	0.91	0.91	**1.00**	0.74	0.74
		Sp	0.63	0.53	0.80	0.84	**0.85**	**1.00**	**0.68**	0.76
mRMR	#10	Ac	0.53	0.65	0.77	0.92	0.89	**1.00**	0.68	0.78
		Se	0.60	**0.95**	0.56	0.96	0.95	0.99	0.74	0.82
		Sp	0.56	0.10	0.90	0.87	0.77	**1.00**	0.62	0.69
	#50	Ac	0.49	0.70	0.84	**0.96**	0.89	**1.00**	0.68	0.79
		Se	0.20	0.77	0.77	**1.00**	0.95	**1.00**	0.74	0.78
		Sp	0.63	0.57	0.87	0.92	0.77	**1.00**	0.62	0.77

(in general) are obtained with the SVM classifier, a subset filter for feature selection, and the DOB-SCV validation method.

4.6 Summary

This chapter has reviewed the up-to-date contributions of feature selection research applied to the field of DNA microarray data analysis. The advent of this type of data has posed a big challenge for machine learning researchers, because of the large input dimensionality and small sample size. Since the infancy of microarray data classification, feature selection has become an imperative step in order to reduce the number of features (genes).

Table 4.16: Experimental results for the SVM classifier on binary datasets after performing DOB-SCV with five folds

			Brain	CNS	Colon	DLBCL	Gli85	Ovarian	Smk	Avg
No FS		Ac	0.67	0.65	0.84	0.93	**0.91**	**1.00**	**0.72**	0.82
		Se	0.30	0.77	**0.82**	0.95	0.97	**1.00**	0.77	0.80
		Sp	0.86	0.44	0.85	0.92	0.77	**1.00**	0.66	**0.79**
CFS		Ac	**0.80**	0.66	0.82	0.94	0.89	**1.00**	0.68	**0.83**
		Se	0.70	0.74	0.78	**1.00**	0.97	0.99	0.70	0.84
		Sp	0.87	0.54	0.85	0.88	0.73	**1.00**	0.66	**0.79**
FCBF		Ac	0.67	0.58	0.79	0.92	0.90	0.99	0.66	0.79
		Se	0.00	0.71	0.68	**1.00**	0.93	0.98	0.68	0.71
		Sp	**1.00**	0.35	0.85	0.84	**0.81**	**1.00**	0.63	0.78
INT		Ac	**0.80**	0.60	0.77	0.91	0.87	**1.00**	**0.72**	0.81
		Se	0.70	0.66	0.63	0.95	0.91	0.99	0.74	0.80
		Sp	0.87	0.49	0.85	0.88	0.77	**1.00**	0.68	**0.79**
IG	#10	Ac	0.62	0.65	0.79	0.91	**0.91**	0.96	0.68	0.79
		Se	0.00	0.81	0.58	0.96	**0.98**	0.94	0.67	0.71
		Sp	0.93	0.34	0.90	0.87	0.73	0.97	0.70	0.78
	#50	Ac	0.66	0.67	**0.85**	**0.98**	0.87	**1.00**	0.65	0.81
		Se	**0.80**	0.74	**0.82**	**1.00**	0.93	**1.00**	0.70	**0.86**
		Sp	0.63	0.54	0.87	**0.96**	0.73	**1.00**	0.60	0.76
ReliefF	#10	Ac	0.45	0.63	0.82	0.95	0.84	0.98	0.65	0.76
		Se	0.10	0.84	0.58	**1.00**	0.93	0.93	0.75	0.73
		Sp	0.60	0.25	**0.95**	0.91	0.62	**1.00**	0.54	0.70
	#50	Ac	0.64	0.63	0.84	0.94	0.88	0.99	**0.72**	0.81
		Se	0.30	0.74	0.81	**1.00**	0.95	0.98	**0.83**	0.80
		Sp	0.80	0.43	0.85	0.88	0.73	**1.00**	0.58	0.75
SVM-RFE	#10	Ac	0.62	0.67	0.76	0.94	0.85	**1.00**	0.67	0.79
		Se	0.10	0.71	0.60	**1.00**	0.95	**1.00**	0.74	0.73
		Sp	0.86	0.58	0.85	0.88	0.61	**1.00**	0.60	0.77
	#50	Ac	0.67	0.65	0.81	0.91	0.86	**1.00**	0.71	0.80
		Se	0.30	0.72	0.78	0.95	0.91	**1.00**	0.70	0.77
		Sp	0.86	0.54	0.82	0.88	0.73	**1.00**	**0.71**	**0.79**
mRMR	#10	Ac	0.45	0.65	0.82	**0.98**	0.87	**1.00**	0.71	0.78
		Se	0.20	**0.87**	0.72	**1.00**	0.95	**1.00**	0.74	0.78
		Sp	0.60	0.24	0.87	**0.96**	0.69	**1.00**	0.66	0.72
	#50	Ac	0.58	**0.70**	0.84	0.93	0.87	**1.00**	0.66	0.80
		Se	0.10	0.74	0.78	0.95	0.95	**1.00**	0.73	0.75
		Sp	0.80	**0.63**	0.87	0.92	0.70	**1.00**	0.57	0.78

Since the end of the 1990s, when microarray datasets began to be dealt with, a large number of feature selection methods have been applied. In the literature one can find both classical methods and methods developed especially for this kind of data. Due to the high computational resources that these datasets demand, wrapper and embedded methods have been mostly avoided, in favor of less expensive approaches such as filters.

The key point to understanding all the attention devoted to microarray data is the challenge that their problem poses. Besides the obvious disadvantage of having so many features for such a small number of samples, researchers also have to deal with classes which are very unbalanced, training and test datasets extracted under different conditions, dataset shift or the presence of outliers. This is why new methods emerge every year, not only to improve previous results in terms of classification

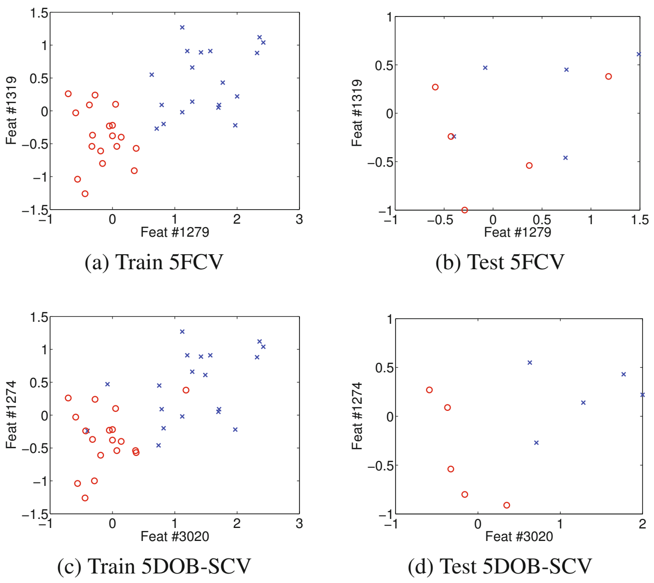


Fig. 4.4: Two first features selected by mRMR in the first fold for both 5-fold cross-validation and 5DOB-SCV

accuracy, but also to help biologists identify the underlying mechanism that relates gene expression to diseases.

The objective of this chapter is not in any way to evaluate feature selection methods for microarray data in terms of which one is the best, but to gather as much up-to-date knowledge as possible for the interested reader. Bearing this in mind, the recent literature has been analyzed in order to give the reader a sense of the approach in developing feature selection methods for microarray data. In order to have a complete picture on the topic, the most common validation techniques have also been mentioned. Since there is no consensus in the literature about this issue, some guidelines have been provided.

Finally, a framework for feature selection evaluation in microarray datasets and a practical evaluation have been provided, where the results obtained have been analyzed. This experimental study tries to show in practice the problems explained in theory. For this sake, a suite of nine widely used binary datasets was chosen to have seven classical feature selection methods applied over them. To obtain the final classification accuracy, three well-known classifiers were used. This large set of experiments aims also at facilitating future comparative studies when a researcher proposes a new method.

Regarding the opportunities for future feature selection research, there tends to be a focus on new combinations such as hybrid or ensemble methods. These types of methods are able to enhance the robustness of the final subset of selected features, which is also a trending topic in this domain. Another interesting line of future research might be to distribute the microarray data vertically (i.e., by features) in order to reduce the heavy computational burden when applying wrapper methods.

References

1. Piatetsky-Shapiro, G. and Tamayo, P. Microarray data mining: facing the challenges. *ACM SIGKDD Explorations Newsletter*, 5(2):1–5, 2003.
2. Saeys, Y., Inza, I. and Larrañaga, P. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.
3. Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A. and others. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, 1999.
4. Jain, A. and Zongker, D. Feature selection: Evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–158, 1997.
5. Garcia Moreno-Torres, J., Sáez, J.A. and Herrera, F. Study on the impact of partition-induced dataset shift on *k*-fold cross-validation. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1304–1312, 2012.
6. Dudoit, S., Fridlyand, J. and Speed, T. P. Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, 97(457):77–87, 2002.
7. Li, T., Zhang, C. and Ogihara, M. A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 20(15):2429–2437, 2004.
8. Lee, J. W., Lee, J. B., Park, M. and Song, S. H. An extensive comparison of recent classification tools applied to microarray data. *Computational Statistics & Data Analysis*, 48(4):869–885, 2005.
9. Ding, C. and Peng, H. Minimum redundancy feature selection from microarray gene expression data. *Journal of Bioinformatics and Computational Biology*, 3(02):185–205, 2005.
10. Yeung, K. Y. and Bumgarner, R. E. Multiclass classification of microarray data with repeated measurements: application to cancer. *Genome Biology*, 4(12):83–83, 2003.
11. Wang, Y., Tetko, I.V., Hall, M. A., Frank, E., Facius, A., Mayer, K. F. X. and Mewes, H. W. Gene selection from microarray data for cancer classification: a machine learning approach. *Computational Biology and Chemistry*, 29(1):37–46, 2005.
12. Gevaert, O., Smet, F., Timmerman, D., Moreau, Y. and Moor, B. Predicting the prognosis of breast cancer by integrating clinical and microarray data with Bayesian networks. *Bioinformatics*, 22(14):184–190, 2006.
13. Blanco, R., Larrañaga, P., Inza, I. and Sierra, B. Gene selection for cancer classification using wrapper approaches. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(08):1373–1390, 2004.
14. Jirapech-Umpai, T. and Aitken, S. Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes. *BMC Bioinformatics*, 6(1):148, 2005.
15. Inza, I., Larrañaga, P., Blanco, R. and Cerrolaza, A. J. Filter versus wrapper gene selection approaches in DNA microarray domains. *Artificial Intelligence in Medicine*, 31(2):91–103, 2004.
16. Ruiz, R., Riquelme, J. C. and Aguilar-Ruiz, J. S. Incremental wrapper-based gene selection from microarray data for cancer classification. *Pattern Recognition*, 39(12):2383–2392, 2006.

17. Dougherty, E. R. Small sample issues for microarray-based classification. *Comparative and Functional Genomics*, 2(1):28–34, 2001.
18. Braga-Neto, U. Fads and fallacies in the name of small-sample microarray classification - A highlight of misunderstanding and erroneous usage in the applications of genomic signal processing. *IEEE Signal Processing Magazine*, 24(1):91–99, 2007.
19. Michiels, S., Koscielny, S. and Hill, C. Prediction of cancer outcome with microarrays: A multiple random validation strategy. *The Lancet*, 365(9458):488–492, 2005.
20. He, H. and Garcia, E. A. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.
21. Sun, Y., Wong, A. K. C. and Kamel, M. S. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(04):687–719, 2009.
22. López, V., Fernández, A., García, S., Palade, V. and Herrera, F. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250(0):113–141, 2013.
23. Galar, M., Fernandez, A., Barrenechea, E., Bustince, H. and Herrera, F. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(4):463–484, 2012.
24. Alizadeh, A. A., Eisen, M. B., Davis, R. E., Ma, C., Lossos, I. S., Rosenwald, A., Boldrick, J. C., Sabet, H., Tran, T., Yu, X. and others. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511, 2000.
25. Chawla, N. V., Bowyer, K. W., Hall, L. O. and Kegelmeyer, W. P. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.
26. Blagus, R. and Lusa, L. Evaluation of SMOTE for high-dimensional class-imbalanced microarray data. In *11th IEEE International Conference on Machine Learning and Applications*, volume 2, pages 89–94, 2012.
27. Galar, M., Fernández, A., Barrenechea, E. and Herrera, F. EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. *Pattern Recognition*, 46(12):3460–3471, 2013.
28. Saez, J. A., Luengo, J. and Herrera, F. Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification. *Pattern Recognition*, 46:355–364, 2013.
29. Ho, T. K. and Basu, M. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):289–300, 2002.
30. Lorena, A. C., Costa, I. G., Spolaôr, N. and de Souto, M. C. P. Analysis of complexity indices for classification problems: cancer gene expression data. *Neurocomputing*, 75(1):33–42, 2012.
31. Okun, O. and Priisalu, H. Dataset complexity in gene expression based cancer classification using ensembles of *k*-nearest neighbors. *Artificial Intelligence in Medicine*, 45(2):151–162, 2009.
32. Moreno-Torres, J. G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N. V. and Herrera, F. A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530, 2012.
33. Gordon, G. J., Jensen, R. V., Hsiao, L-L. and Gullans, S. R., Blumenstock, J. E., Ramaswamy, S., Richards, W. G., Sugarbaker, D. J. and Bueno, R. Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. *Cancer Research*, 62(17):4963–4967, 2002.
34. Singh, D., Febbo, P. G., Ross, K., Jackson, D. G., Manola, J., Ladd, C., Tamayo, P., Renshaw, A. A., D'Amico, A. V., Richie, J. P. and others. Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, 1(2):203–209, 2002.
35. Bolón-Canedo, V., Sánchez-Maroño, N. and Alonso-Betanzos, A. On the effectiveness of discretization on gene selection of microarray data. In *IEEE International Joint Conference on Neural Networks*, pages 18–23, 2010.
36. Bolón-Canedo, V., Sánchez-Maroño, N. and Alonso-Betanzos, A. Data classification using an ensemble of filters. *Neurocomputing*, 135:13–20, 2014.
37. Barnett, V. and Lewis, T. *Outliers in Statistical Data*, volume 3. Wiley, New York, 1994.

38. Kadota, K., Tominaga, D., Akiyama, Y. and Takahashi, K. Detecting outlying samples in microarray data: A critical assessment of the effect of outliers on sample classification. *Chem-Bio Informatics*, 3(1):30–45, 2003.
39. Gonzalez-Navarro, F. F. *Feature Selection in Cancer Research: Microarray Gene Expression and in Vivo 1H-MRS Domains*. Ph.D. Thesis, Technical University of Catalonia, 2011.
40. Wang, J., Wu, L., Kong, J., Li, Y. and Zhang, B. Maximum weight and minimum redundancy: A novel framework for feature subset selection. *Pattern Recognition*, 46(6):1616–1627, 2013.
41. Bolón-Canedo, V., Seth, S., Sánchez-Maroño, N., Alonso-Betanzos, A. and Principe, J. C. Statistical dependence measure for feature selection in microarray datasets. In *19th European Symposium on Artificial Neural Networks*, pages 23–28, 2011.
42. Meyer, P. E., Schretter, C. and Bontempi, G. Information-theoretic feature selection in microarray data using variable complementarity. *IEEE Journal of Selected Topics in Signal Processing*, 2(3):261–274, 2008.
43. González Navarro, F. F. and Belanche Muñoz, L. A. Gene subset selection in microarray data using entropic filtering for cancer classification. *Expert Systems*, 26(1):113–124, 2009.
44. Song, L., Smola, A., Gretton, A., Bedo, J. and Borgwardt, K. Feature selection via dependence maximization. *The Journal of Machine Learning Research*, 13:1393–1434, 2012.
45. García, S., Luengo, J., Sáez, J. A., López, V. and Herrera, F. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, 25(4), 2013.
46. Ferreira, A. J. and Figueiredo, M. A. T. An unsupervised approach to feature discretization and selection. *Pattern Recognition*, 45(9):3048–3060, 2012.
47. Sánchez-Maroño, N., Alonso-Betanzos, A., García-González, P. and Bolón-Canedo, V. Multiclass classifiers vs. multiple binary classifiers using filters for feature selection. In *IEEE International Joint Conference on Neural Networks*, pages 18–23, 2010.
48. Nie, F., Huang, H., Cai, X. and Ding, C. Efficient and robust feature selection via joint l2, 1-norms minimization. *Advances in Neural Information Processing Systems*, 23:1813–1821, 2010.
49. Lan, L. and Vucetic, S. Improving accuracy of microarray classification by a simple multi-task feature selection filter. *International Journal of Data Mining and Bioinformatics*, 5(2):189–208, 2011.
50. Student, S. and Fujarewicz, K. Stable feature selection and classification algorithms for multiclass microarray data. *Biology Direct*, 7(1):33, 2012.
51. Inza, I., Sierra, B., Blanco, R. and Larrañaga, P. Gene selection by sequential search wrapper approaches in microarray cancer class prediction. *Journal of Intelligent and Fuzzy Systems*, 12(1):25–33, 2002.
52. Sharma, A., Imoto, S. and Miyano, S. A top-r feature selection algorithm for microarray gene expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(3):754–764, 2012.
53. Wanderley, M. F., Gardeux, V., Natowicz, R. and Braga, A. P. GA-KDE-Bayes: An evolutionary wrapper method based on non-parametric density estimation applied to bioinformatics problems. In *21st European Symposium on Artificial Neural Networks*, pages 155–160, 2013.
54. Guyon, I., Weston, J., Barnhill, S. and Vapnik, V. Gene selection for cancer classification using Support Vector Machines. *Machine Learning*, 46(1-3):389–422, 2002.
55. Maldonado, S., Weber, R. and Basak, J. Simultaneous feature selection and classification using kernel-penalized support vector machines. *Information Sciences*, 181(1):115–128, 2011.
56. Wang, G., Song, Q., Xu, B. and Zhou, Y. Selecting feature subset for high dimensional data via the propositional foil rules. *Pattern Recognition*, 46(1):199–214, 2013.
57. Shah, M., Marchand, M. and Corbeil, J. Feature selection with conjunctions of decision stumps and learning from microarray data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):174–186, 2012.
58. Canul-Reich, J., Hall, L. O., Goldgof, D. B., Korecki, J. N. and Eschrich, S. Iterative feature perturbation as a gene selector for microarray data. *International Journal of Pattern Recognition and Artificial Intelligence*, 26(05), 2012.

59. Anaissi, A., Kennedy, P. J. and Goyal, M. Feature selection of imbalanced gene expression microarray data. In *12th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pages 73–78. IEEE, 2011.
60. Mundra, P. A. and Rajapakse, J. C. SVM-RFE with mRMR filter for gene selection. *IEEE Transactions on NanoBioscience*, 9(1):31–37, 2010.
61. Shreem, S. S., Abdullah, S., Nazri, M. Z. A. and Alzaqebah, M. Hybridizing ReliefF, MRMR filters and GA wrapper approaches for gene selection. *Journal of Theoretical and Applied Information Technology*, 46(2):1034–1039, 2012.
62. Chuang, L., Yang, C., Wu, K. and Yang, C. A hybrid feature selection method for DNA microarray data. *Computers in Biology and Medicine*, 41(4):228–237, 2011.
63. Lee, C. and Leu, Y. A novel hybrid feature selection method for microarray data analysis. *Applied Soft Computing*, 11(1):208–213, 2011.
64. Leung, Y. and Hung, Y. A multiple-filter-multiple-wrapper approach to gene selection and microarray data classification. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(1):108–117, 2010.
65. Bolón-Canedo, V., Sánchez-Maroño, N. and Alonso-Betanzos, A. An ensemble of filters and classifiers for microarray data classification. *Pattern Recognition*, 45(1):531–539, 2012.
66. Yang, F. and Mao, K. Z. Robust feature selection for microarray data based on multicriterion fusion. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(4):1080–1092, 2011.
67. Abeel, T., Helleputte, T., Van de Peer, Y., Dupont, P. and Saeys, Y. Robust biomarker identification for cancer diagnosis with ensemble feature selection methods. *Bioinformatics*, 26(3):392–398, 2010.
68. Ye, Y., Wu, Q., Zhexue Huang, J., Ng, M. K. and Li, X. Stratified sampling for feature subspace selection in random forests for high dimensional data. *Pattern Recognition*, 46(3):769–787, 2013.
69. Lovato, P., Bicego, M., Cristani, M., Jojic, N. and Perina, A. Feature selection using counting grids: application to microarray data. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 629–637. Springer, 2012.
70. Song, A., Ni, J. and Wang, G. A fast clustering-based feature subset selection algorithm for high-dimensional data. *IEEE Transactions on Knowledge and Data Engineering*, 25(1):1–14, 2013.
71. Ambroise, C. and McLachlan, G. J. Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the National Academy of Sciences*, 99(10):6562–6566, 2002.
72. Braga-Neto, U. M. and Dougherty, E. R. Is cross-validation valid for small-sample microarray classification? *Bioinformatics*, 20(3):374–380, 2004.
73. Feature Selection at Arizona State University. http://featureselection.asu.edu/datasets.php
74. Statnikov, A., Aliferis, C. F. and Tsamardinos, I. GEMS: Gene Expression Model Selector http://www.gems-system.org
75. Kent Ridge Bio-Medical Dataset. http://datam.i2r.a-star.edu.sg/datasets/krbd
76. Orriols-Puig, A. and Bernadó-Mansilla, E. Evolutionary rule-based systems for imbalanced data sets. *Soft Computing*, 13(3):213–225, 2009.
77. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I. H. The Weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

Chapter 5
Application of Feature Selection to Real Problems

Abstract This chapter presents two real problems where feature selection has proved to be useful in improving performance. Section 5.1 is devoted to improving the classification accuracy of the KDD Cup 99 dataset, a benchmark in the intrusion detection field. A method based on the combination of discretization, filtering and classification algorithms is proposed, to be applied to both the binary and the multiclass version of the dataset. Then, the second problem is presented in Section 5.2, which is related to tear film lipid layer classification. A methodology making use of feature selection methods is proposed, achieving considerable improvements in performance. Since the second problem has the added handicap of having to reduce the cost associated with the features, in Section 5.3 we introduce a general framework for cost-based feature selection. Section 5.4 closes this chapter with a summary of its findings.

This chapter is devoted to proving the benefits of feature selection in other real applications apart from DNA microarray data. Feature selection may be very useful in real domains, since it allows the storage costs to decrease, the performance of a classifier to improve and a good understanding of the model to be obtained. Two cases when feature selection has reported success will be presented: classification of intrusion detection systems and classification of the tear film lipid layer. The latter has the added problem of having to reduce the cost associated with the features. For this reason, in the last section of this chapter we present a framework for cost-based feature selection.

Part of the content of this chapter was previously published in *Expert Systems with Applications* (`doi:10.1016/j.eswa.2010.11.028`) and in the *IEEE Journal of Biomedical and Health Informatics* (`doi:10.1109/JBHI.2013.2294732`)

V. Bolón-Canedo et al., *Feature Selection for High-Dimensional Data*, Artificial Intelligence: Foundations, Theory, and Algorithms, DOI 10.1007/978-3-319-21858-8_5

5.1 Classification in Intrusion Detection Systems

Intrusion detection is a classification task that is getting great attention in the machine learning community because of the continuous increase of attacks in computer networks. Therefore, it is necessary to build effective techniques to protect the information systems. This classification task consists of building a predictive model (i.e., a classifier) capable of distinguishing between intrusions or attacks and normal connections. Then, it can be restricted to considering a two-class problem, focused on distinguishing attacks or normal patterns, or a multiple class problem, dealing with the classification of the different attacks suffered. In both cases, the goal is to produce a classifier that will work correctly on unseen examples (test dataset), i.e., it generalizes well.

In complex classification domains, such as intrusion detection systems (IDSs), features or attributes may contain false correlations, which hinder the underlying process and, in general, the learning task to be carried out. Furthermore, some features may be irrelevant and some others may be redundant since the information they add is contained in other features. For these reasons, these types of datasets are good candidates for performing feature selection.

A benchmark dataset in the intrusion detection field is the KDD (Knowledge Discovery and Data Mining Tools Conference) Cup 99 dataset [1] (employed in the KDD Cup 99 competition), which contains five million samples represented by employing 41 features or attributes. This database contains 39 types of distinct attacks, grouped into four classes of attack and one class of non-attack. Also, in the test set there are new attacks that are not present in the training set (this is another example of dataset shift; see Chapter 1). These characteristics convert this dataset into a challenge for the sake of classification. Moreover, there are some features with very unbalanced values (in both training and test sets) and some of them are constant in the training set provided by the competition organizers. The dataset that will be used in our study is a smaller subset (10% of the original training set) that contains 494,021 instances and was already employed as the training set in the competition. For the test set, we used the original KDD Cup 99 dataset containing 331,029 patterns. Around 20% of the two datasets are normal patterns (no attacks). As for attack patterns, the 39 types of attacks are grouped into four categories [2]:

- Denial of Service (DoS) attacks, where an attacker makes some computing or memory resource too busy or too full to handle legitimate requests, thus denying legitimate users access to a machine.
- Probe attacks, where an attacker scans a network to gather information or find known vulnerabilities.
- Remote-to-Local (R2L) attacks, where an attacker sends packets to a machine over a network, then exploits the machine's vulnerability to illegally gain local access as a user.
- User-to-Root (U2R) attacks, where an attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain root access to the system.

The training and test set percentages for normal activities and for the four attack types are shown in Table 5.1. The KDD Cup 99 problem can be treated under two approaches: the binary case, which consists of distinguishing between attack and no attack, and the multiple class case, which consists of distinguishing the kind of attack it is.

Table 5.1: Percentages of distribution of normal activities and different kinds of attacks in the KDD Cup 99

Type	% Train Set	% Test Set
Normal	19.69	19.48
DoS	79.24	73.90
Probe	0.83	1.34
R2L	0.23	5.21
U2R	0.01	0.07

As Table 5.1 shows, the percentage of attacks in both datasets is very high, more than 80%, where most of the attacks belong to type DoS. Furthermore, it is a very unbalanced dataset, with some classes (such as U2R and R2L) containing very few samples, which will make their classification difficult during the learning stage.

Table 5.2: Unbalanced continuous attributes of the KDD Cup 99 dataset

Feature	Min.	Max.	Mean	StdDev	Distinct
duration	0	58 329	47.98	707.75	2495
src_bytes	0	693 375 640	3025.61	988 218.10	3300
dst_bytes	0	5 155 468	868.53	33 040.00	10 725

The KDD Cup 99 training dataset (494,021 instances) is a good candidate for feature selection because of the characteristics of its input attributes. There are two features that are constant (*num_outbound_cmds* and *is_host_login*) and some that are almost constant (*land*, *root_shell*, *num_shells*). Apart from constant features, the KDD Cup 99 dataset has continuous features that are very skewed and for which a possible solution could be discretizing numeric data. Some examples of these attributes can be viewed in Table 5.2, where the minimum and maximum value of each feature is shown, as well as its mean, standard deviation and the number of distinct examples.

In order to solve the aforementioned problems, a three-step methodology is proposed for both binary and multiclass classification.

- First, a discretizer method is applied over the input data, with the aim of solving problems of unbalanced values, and preparing the attributes of the sample to be

processed by the feature selection algorithm from the next step. Several discretizers have been chosen to test for their influence on the classification problem.
- After discretization, feature selection is carried out using filters.
- Finally, a classifier is applied.

Many discretization methods exist in the literature, but in this research only the most suitable for large datasets have been chosen [3]. In this manner, EMD (Entropy Minimization Discretization) [4] was chosen because it is a classic algorithm, and PKID [5] because it is a new approach that works well with large datasets. Both of them are described in Appendix A.

As each combination of discretizer and filter may lead to different performance results depending on the classifier used, several classifiers have to be checked. Many datasets, such as the KDD Cup 99, contain different types of attributes (symbolic, numerical, binary) and it is well known that not all classifiers may deal with symbolic attributes. Therefore, in some cases, a symbolic conversion is required. In these experiments, different classifiers will be used. Some of them do not need the symbolic conversion (C4.5 or naive Bayes) whereas others do demand it (one-layer NN, PSVM, MLP). A detailed description of all of them is available in Appendix A, Section A.6.

5.1.1 Results on the Binary Case

In general, the users of an intrusion detection system (IDS) are interested in differentiating between normal connections and attack situations in first stage. Later, they would like to distinguish the type of attack that they have suffered. In this sense, the KDD Cup 99 dataset can be considered as a binary problem, detecting normal versus attack patterns, or a multiple class problem, classifying different types of attacks.

The goal of this research is to reduce the number of features of the KDD Cup 99 dataset in both cases, binary and multiple class, but maintain the performance results. Therefore, the proposed method, consisting of a combination of discretizer, filter and classifier, was applied first, for the sake of simplicity, to the binary case. Later, it will be applied to the multiple class case.

The distribution of the classes for the binary case can be seen in Table 5.3. Classes can be checked to be clearly unbalanced, although, contrary to the microarray datasets presented in Chapter 4, enough samples of the minority class exist (97,278) for the adequate training of the classifiers.

Table 5.4 illustrates the results achieved in a previous work by the KDD Cup winner and other authors in the literature [6] using the whole set of features (41). The performance measures used are the error (percentage of samples incorrectly classified), true positive rate and false positive rate (see Appendix A, Section A.7).

Different filters (CFS, consistency-based and INTERACT; see Chapter 2) were applied to achieve a feature reduction. Moreover, some filters require a discretization step. So, different analyses were carried out to check the influence of the distinct

Table 5.3: Percentages of distribution of normal activities and attacks in the KDD Cup 99 training and test datasets for the binary case

Type	% Train Set	% Test set
Normal	19.69	19.48
Attack	80.31	80.52

Table 5.4: Results obtained in the binary case over the test set by other authors (in %)

Method	Error	True Positives	False Positives
KDD winner	6.70	91.80	0.55
5FNs_poly	6.48	92.45	0.86
5FNs_fourier	6.69	92.72	0.75
5FNs_exp	6.70	92.75	0.75
SVM Linear	6.89	91.83	1.62
SVM 2poly	6.95	91.79	1.74
SVM 3poly	7.10	91.67	1.94
SVM RBF	6.86	91.83	1.43
ANOVA ens.	6.88	91.67	0.90
Pocket 2cl.	6.90	91.80	1.52
Pocket mcl.	6.93	91.86	1.96

discretizators and filters. The results obtained show that both discretizators and filters have a great influence in the results of the classifiers, as can be seen in a previous work [6]. In fact, each combination of discretizator and filter leads to a different subset of features. Therefore, it was not possible to determine which combination was the best and several combinations were tested over the KDD Cup 99 binary dataset.

The classifier stage also affects the performance results; therefore it is necessary to check several classifiers. Two types of classifiers will be tested: those which can deal with symbolic attributes and those which cannot. For the latter, note that a symbolic conversion will be required. The next sections will described the experiments related with this issue.

5.1.1.1 Classifiers Without Symbolic Conversion

This subsection shows the results obtained with two classifiers that can deal with both numerical and symbolic attributes so that no conversion is needed: C4.5 and naive Bayes (see Appendix A). In the case of C4.5, the confidence factor parameter was tuned in order to obtain the best results, which have been set to 0.25 and 0.50.

Table 5.5 shows the best results obtained in a previous work [6], where C4.5 results outperform the KDD winner score. However, in general, none of the results obtained with naive Bayes improved it and thus only the best result achieved is included. The measures employed are again the error, true positive's rate and false positive's rate, computed over the same test set, as well as the number of features used. The method applied achieved good results over the binary case, outperforming the KDD winner results in terms of error and true positive rate, while false positive rate maintains reasonable values (see Table 5.4) with only 17% of the total features. The result obtained with the combinaton EMD+INT+C4.5(0.50) is remarkable, since none of the other authors (Table 5.4) have achieved better results than the KDD winner in all three measures. It should be mentioned that the FP rate remains low while maintaining the Error and TP rates. Note that the FP rate is a measure of immense importance in determining the quality of an IDS system [7].

Table 5.5: Results obtained in the binary case over the test set (in %)

Method	Error	True Positives	False Positives	No. of Features
PKID+Cons+C4.5(0.25)	5.15	94.07	1.90	6
PKID+Cons+C4.5(0.50)	**5.14**	**94.08**	1.92	6
EMD+INT+C4.5(0.25)	6.74	91.73	0.44	7
EMD+INT+C4.5(0.50)	6.69	91.81	0.49	7
PKID+Cons+NB	7.99	90.18	**0.42**	6

5.1.1.2 Classifiers with Symbolic Conversion

Another three classifiers have been tested over the binary KDD Cup 99: one-layer NN, PSVM and MLP (see Appendix A). One-layer NN and PSVM were chosen due to their good performance and reduced time costs while MLP was selected because it is one of the most employed neural network architectures. Unlike the methods tested before (C4.5 and naive Bayes), these classifiers need a previous conversion of the symbolic attributes of the dataset into numerical labels. For this task, the Separability of Split Value (SSV) criterion-based method [8] was chosen due to the fact that it is a successful tool for symbolic to real-valued feature mapping.

The results obtained with these methods following the process illustrated in Figure 5.1 (i.e., only the symbolic features selected by filters are converted) are shown in Table 5.6 and compared with the best results obtained in [6] with C4.5 and naive Bayes.

As can be seen, the results obtained with the three first classifiers of Table 5.6 did not improve those results shown in the previous subsection with naive Bayes and, especially, with C4.5 (see Table 5.5). Hence, based on these results, it seems better to use classifiers that do not require symbolic conversion methods, which in fact might cause information loss.

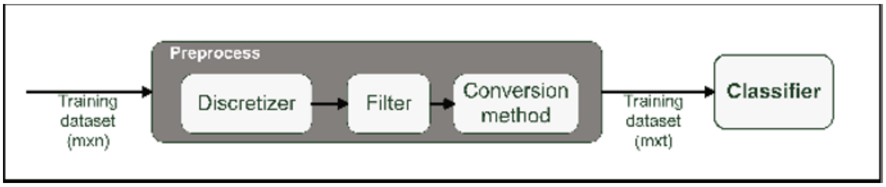


Fig. 5.1: Preprocessing for the classifiers with symbolic conversion

Table 5.6: Comparison of the results (in %) obtained in the binary case with several classifiers over the test set

Method	Error	True Positives	False Positives	No. of Features
EMD+Cons+One-layer	7.78	90.52	0.77	6
EMD+Cons+PSVM	7.78	90.53	0.78	6
EMD+Cons+MLP	8.01	90.18	0.54	6
EMD+INT+C4.5	6.69	91.81	0.49	7
PKID+Cons+NB	7.99	90.18	0.42	6

5.1.2 Results on the Multiple Class Case

The simplest way to classify a dataset with more than two classes is to use a multiple class classifier. However, it is not a very extended choice, because not all the machine learning algorithms have this capacity. C4.5 and naive Bayes classifiers have turned out good results in the binary case and they can deal with multiple classes. This approach, however, has the risk of focusing on the majority classes (see Table 5.1) and good results are not expected [9].

Therefore, a more common approach is the one based on multiple binary classifiers. This approach consists of employing class binarization techniques which reduce the multiple class problem into a series of binary problems which are solved individually. Then, the resultant predictions are combined to obtain a final solution [10]. There are various class binarization approaches proposed in the literature, and all of them can be summarized using a coding matrix $M_{b\times c}$, b being the number of classifiers and c the number of classes. A simple example for a problem with four classes is shown in Figure 5.2. White and black boxes are positive and negative examples, respectively, while gray boxes represent samples that must be ignored for the corresponding classifier.

Once each classifier is individually trained, the global performance must be checked using a test dataset. Then, a test sample x is fed to each learning algorithm that finds a hypothesis for it. The vector of predictions of these classifiers on an instance x is denoted as $\mathbf{h}(\mathbf{x}) = (h_1(x),\ldots,h_b(x))$ and the yth column of the matrix $M_{b\times c}$ is denoted by M_y. Given an instance x, the label y for which the column M_y is the *closest* to $\mathbf{h}(\mathbf{x})$ is predicted. In other words, the class y is predicted to

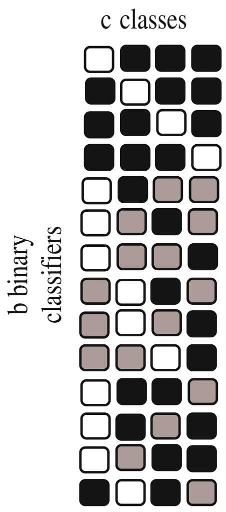


Fig. 5.2: Code matrix for a four-class problem

minimize the $d(M(y), \mathbf{h}(\mathbf{x}))$ for some distance d. Several distance measures can be considered, leading to different results.

Several class binarization techniques [10] have been proposed in the literature. This research, however, is focused on the two more popular ones, namely *One vs. Rest* and *One vs. One*.

- **One vs. Rest (OvR):** This is the most popular class binarization technique, where one takes each class in turn and learns binary concepts that distinguish that class from all other classes. This technique transforms a c-class problem into c two-class problems. These two-class problems are constructed by using the examples of class i as the positive examples and the examples of the rest of the classes as the negative examples. Notice that this technique corresponds to the first four rows of Figure 5.2.
- **One vs. One (OvO):** This class binarization technique consists of learning one classifier for each pair of classes. The *One vs. One* class binarization transforms a c-class problem into $\frac{c(c-1)}{2}$ two-class problems, one for each set of classes $\{i, j\}$. The binary classifier for a problem is trained with examples of its corresponding classes i, j, whereas examples of the rest of the classes are ignored for this problem. This technique is represented by rows five to ten in Figure 5.2.

A c-class problem is subdivided into b binary problems according to the matrix $M_{b\times c}$, and then each problem is solved by using a different classifier that is trained individually. Given a new sample x, one classifier obtains a probability (p) of assigning that pattern to a given class and the rest $(1-p)$ to the opposite class. However, in some cases, a given test sample may not be assigned to any of those classes, because the classifier was not trained with them (consider for example a pattern of class 3 or 4 in the fifth classifier (fifth row) in Figure 5.2). Therefore, there are three possible outputs: positive ($+1$), negative (-1) or ignored (0). Then, to assign a pattern to the positive or negative classes, their corresponding probability must be higher than a determined threshold ($p > \sigma$); otherwise the considered output is *ignored*. Given a test pattern x, each individual classifier will obtain its corresponding output, $\mathbf{h}(\mathbf{x}) = (h_1(x), \ldots, h_b(x))$. However, a global result must be finally reached. Different techniques exist to obtain the proper output, two of them use a measure distance between the vector $\mathbf{h}(\mathbf{x})$ and the columns of matrix $M_{b\times c}$, M_y and they are subsequently illustrated. Finally, three different measures were designed ad hoc based on the probabilities.

- **Hamming decoding:** This technique [11] counts the number of positions in which the sign of the vector of predictions $\mathbf{h}(\mathbf{x})$ differs from the matrix column M_y.
- **Loss-based decoding:** Unlike Hamming decoding, this method [11] takes into account the magnitude of the predictions, which can often be an indication of a level of *confidence* as well as the relevant *loss* function L. The idea is to choose the label y that is most consistent with the predictions $h_s(x)$ in the sense that, if instance x were labeled y, the total *loss* on instance (x,y) would be minimized over choices of $y \in \{1,\ldots,c\}$.
- **Accumulative sum:** Considering the learning methods used (NB and C4.5), for each sample, the binary classifier obtains a probability (p) for the winning class (positive or negative) while the probability for the remaining class is $1-p$. Therefore, instead of calculating distances to determine the class from the b different results, the accumulative probability sum of each class is computed. Then, the desired output is the one with the highest value.
- **Accumulative sum with threshold:** This method is a modification of the previous one and takes under consideration the fact that test patterns include *ignored* classes, i.e., classes not used for the learning of the classifier. Then, this technique only computes those probabilities that are over an established threshold to guarantee that only clearly winning classes are computed.
- **Probability accumulative sum:** Previous techniques are more adequate for *One vs. One* binarization because of the existence of *ignored* outputs. In the *One vs. Rest* approach, a different technique was employed. Notice that, if all the classifiers work properly, each pattern of a given class will be recognized by the classifier of the class, while it will be assigned to class *rest* by the remaining classifiers. However, if one of the classifiers does not work well, a conflict appears, and the same pattern can be assigned to several classes, one for each classifier. Again, the cumulative sum of probabilities was selected to determine the adequate class.

The aim of the experiments that will be presented in this section is to overcome the KDD winner score, which was measured according to the cost matrix that is shown in Table 5.7 [12]. The KDD winner obtained a score of 0.2331 and non-winning entries obtained an average cost per test example ranging from 0.2356 to 0.9414.

Table 5.7: Cost matrix used in the KDD Cup 99 competition

		Prediction				
		Normal	Probe	DoS	U2R	R2L
Real	Normal	0	1	2	2	2
	Probe	1	0	2	2	2
	DoS	2	1	0	2	2
	U2R	3	2	2	0	2
	R2L	4	2	2	2	0

As was stated in Table 5.1, the classes are clearly unbalanced in both training and test sets. Normal and DoS classes have enough samples, while the number of instances available for U2R and R2L is small and, moreover, different from that in the test set. In fact, they may suffer from the data shift phenomenon (see Chapters 1 and 4). Then, some classes are clearly difficult to learn, and for that reason it is also necessary to compute the detection for each class (true positives).

Table 5.8 displays the best results for the multiple class case, in which the best score is highlighted in boldface. Notice that the score is calculated according to the competition cost matrix seen in Table 5.7, and the lower the score, the better the behavior. The first row shows the result obtained by the KDD Winner, the second row displays the best result obtained by a multiple class algorithm (Multi) and the third row reports the result for the *One vs. Rest* technique (OvR). Finally, the last four rows reveal the best results for the *One vs. One* approach (OvO) combined with the four different decoding techniques.

Table 5.8: Best test results obtained with the multiple class algorithms

Combination	Decoding technique	Score	No. of features
KDD winner	–	0.2331	41
EMD + INT + C4.5 (Multi)	–	0.2344	11
EMD + INT + C4.5 (OvR)	Accumulative sum	0.2324	15
PKID + Cons + C4.5 (OvO)	Sum	**0.2132**	13
PKID + Cons + C4.5 (OvO)	Sum-with-threshold	0.2209	13
PKID + Cons + C4.5 (OvO)	Loss-based	0.2167	13
PKID + Cons + C4.5 (OvO)	Hamming	0.2242	13

For the multiclass algorithm, several combinations of discretizators, filters and classifiers were tested. A total of 13 experiments were conducted, but none of them improved the KDD winner score. However, the best result achieved is close to the KDD winner result, but using a significant reduction in the number of features (see Table 5.8). As for the results achieved with the *One vs. Rest* approach (third row of Table 5.8), one of the combinations improved the KDD winner score with an important reduction (37%) of the total number of features.

The results for the *One vs. One* approach shown in Table 5.8 were obtained employing the PKID discretizer, the consistency-based filter and the C4.5 classifier, combined with the four decoding techniques mentioned above. They manage to improve the KDD winner score using only 32% of the total features. The first result is very remarkable, achieved with the *Accumulative Sum* decoding technique, since the difference with the KDD winner score is very wide and exactly the same as the one existing between the winner and the tenth entry of the competition [12].

5.1.2.1 Comparison with Other Authors

Several works exist in the literature [13, 14, 15] that deal with the KDD Cup 99 problem, so in this subsection a comparative study will be carried out. In order to compare our results with those obtained by other authors, it is not possible to use the score employed in the prior comparisons with the KDD winner (according to the cost matrix seen in Table 5.7), due to the fact that the confusion matrix is not available for those results and only the detection percentage of attacks for each class is provided (see Table 5.9). Then, the same performance measure will be used to make fair comparisons, although it is not possible to perform statistical tests in order to determine if there are significant differences.

Table 5.9 shows the detection percentage of each class. The six first rows correspond with results achieved in this study and are compared with those results obtained by the KDD winner and other authors. The best result for each class is emphasized in boldfont.

The first six results were obtained with the proposed methodology presented in previous sections. Comparing our results with those obtained by the KDD winner, our approach improves the detection in the minority classes U2R, R2L and Probe, which are the most difficult classes to detect (see Table 5.1). Regarding the results obtained by other authors, it can be seen again that we achieve the best detections in classes R2L and Probe. Note that, in any case, the number of features used by this approach is considerably lower than those of other authors. This is very important because this reduction in the number of features causes an important decrement in the data processing and the data storage costs besides obtaining an interesting improvement in performance.

5.2 Tear Film Lipid Layer Classification

Dry eye is a symptomatic disease which affects a wide range of the population and has a negative impact on their daily activities. Its diagnosis can be achieved by analyzing the interference patterns of the tear film lipid layer and by classifying

Table 5.9: Best results obtained over the test dataset. Comparison with other authors

Combination	Normal	U2R	DoS	R2L	Probe
PKID + Cons + C4.5	97.08	25.00	96.08	8.12	73.62
EMD + INT + C4.5	96.36	19.30	94.07	**32.87**	79.48
EMD + CFS + C4.5 (1)	96.76	21.05	92.54	26.29	86.08
EMD + CFS + C4.5 (2)	96.56	32.89	93.35	5.32	78.18
PKID + Cons + NB (1)	96.63	24.12	90.20	29.98	89.94
PKID + Cons + NB (2)	96.13	11.40	95.12	13.85	**96.67**
KDD winner	**99.45**	13.16	97.12	8.40	83.32
5FNs_poly	92.45	8.33	96.77	29.43	85.96
5FNs_fourier	92.72	10.97	96.86	23.75	85.74
5FNs_exp	92.75	13.60	96.85	23.77	85.60
SVM Linear	91.83	21.93	97.38	16.55	81.76
SVM 2poly	91.79	1.75	97.41	14.74	86.44
SVM 3poly	91.67	2.63	97.62	9.35	88.45
SVM RBF	91.83	25.88	97.30	18.29	79.26
ANOVA ens.	91.67	53.94	97.64	8.51	87.52
Pocket 2cl.	91.80	29.82	97.40	14.77	85.84
Pocket mcl.	91.86	**54.38**	**97.65**	11.45	86.79

them into one of the Guillon categories [16]: open meshwork, closed meshwork, wave, amorphous and color fringe. However, the classification into these grades is a difficult clinical task, especially with thinner lipid layers that lack color and/or morphological features. The subjective interpretation of the experts via visual inspection may affect the classification. This time-consuming task is very dependent on the training and experience of the optometrist(s), and so produces a high degree of inter- and also intra-observer variability [17]. The development of a systematic and objective computerized method for analysis and classification is thus highly desirable, allowing for homogeneous diagnosis and relieving the experts from this tedious task.

Remeseiro et al. [18] proposed a wide set of feature vectors using different texture analysis methods in three color spaces and a 95% of classification accuracy was obtained. Nevertheless, the problem with this approach is that the time required to extract some of the textural features is too long (more than one minute). Interviews with optometrists revealed that a scale of computation time over 10 seconds per image makes the system unusable. Therefore, the previous approach prevents the practical clinical use of an application developed to automatize the process, because it could not work in real time. So, the objective of the work presented in this section is to obtain a reduction in time that allows for the system to be used in daily clinical routine. In order to deal with this problem, feature selection can play a crucial role. Because the number of features to extract and process will be reduced, the time required will be also reduced in consonance, and most of the time, this can be achieved with a minimum degradation of performance.

In order to obtain an efficient method for automatic tear film lipid layer classification, a five-step methodology is applied as illustrated in Figure 5.3. First, feature extraction is performed, after which feature selection methods are applied to select the subset of relevant features that allow for correct classification of the tear film lipid layer thickness. After that, several performance measures are computed to evaluate the performance of the system. Finally, a multiple-criteria decision-making method is carried out to obtain a final result.

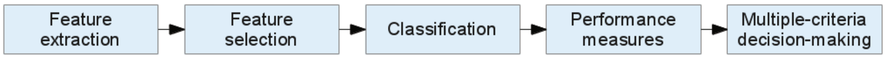


Fig. 5.3: Steps of the research methodology

The initial stage in this methodology is the processing of tear film images in order to extract their features. Firstly, the region of interest of an input image in RGB is extracted. Then, this extracted region in RGB is converted to the Lab color space and its channels L, a and b are obtained. After that, the texture features of each channel are extracted and three individual descriptors are generated. Finally, the three individual descriptors are concatenated in order to generate the final descriptor of the input image, which contains its color and texture features. Five different techniques for texture analysis are tested in this study:

- *Butterworth bandpass filters* [19] are frequency domain filters that have an approximately flat response in the bandpass frequency, which gradually decays in the stopband. The order of the filter defines the slope of the decay; the higher the order, the faster the decay.
- The *discrete wavelet transform* [20] generates a set of wavelets by scaling and translating a *mother wavelet*. The wavelet decomposition of an image consists of applying these wavelets horizontally and vertically, generating four images (LL, LH, HL, HH). The process is repeated iteratively on the LL subimage, resulting in the standard pyramidal wavelet decomposition.
- *Co-occurrence features* method was introduced by [21]; it is based on the computation of the conditional joint probabilities of all pairwise combinations of gray levels. This method generates a set of gray-level co-occurrence matrices and extracts several statistics from their elements. In general, the number of orientations, and so of matrices, for a distance d is $4d$.
- *Markov random fields* [22] generate a texture model by expressing the gray values of each pixel in an image as a function of the gray values in its neighborhood.
- *Gabor filters* [23] are complex exponential functions modulated by Gaussian functions. The parameters of Gabor filters define their shape, and represent their location in the spatial and frequency domains.

Table 5.10 shows the arrangements for applying the texture analysis methods. Note that column *No. of features* contains the number of features generated by each method, in which they are always multiplied by 3 because of the use of Lab.

Table 5.10: Arrangements for texture analysis methods and number of features

Texture analysis	Configuration (per component)	No. features
Butterworth filters	A bank of Butterworth bandpass filters composed of nine second order filters was used, with bandpass frequencies covering the whole frequency spectrum. The filter bank maps each input image to nine output images, one per frequency band. Each output image was normalized separately and then a uniform histogram with non-equidistant bins [24] was computed. Since 16 bin histograms were used, the feature vectors contain 16 components per frequency band.	$9 \times 16 \times 3 = 432$
Discrete wavelet transform	A Haar algorithm [20] was applied as the *mother wavelet*. The descriptor of an input image is constructed calculating the mean and the absolute average deviation of the input and LL images, and the energy of the LH, HL and HH images. Since two scales were used, the feature vectors contain 12 components.	$12 \times 3 = 36$
Co-occurrence features	A set of 14 statistics proposed by [21] was computed from each co-occurrence matrix. These statistics represent features such as homogeneity or contrast. The descriptor of an image consists of two properties, the mean and range across matrices of these statistics, thus obtaining a feature vector with 28 components per distance. Distances from one to seven were considered.	$28 \times 7 \times 3 = 588$
Markov random fields	In this work, the neighborhood of a pixel is defined as the set of pixels within a Chebyshev distance d. To generate the descriptor, the directional variances proposed by [25] were used. For a distance d, the descriptor comprises $4d$ features. Distances from one to ten were considered.	$(\Sigma_{d=1}^{10} 4d) \times 3 = 660$
Gabor filters	A bank of 16 Gabor filters centered at four frequencies and four orientations was created. The filter bank maps each input image to 16 output images, one per frequency-orientation pair. Using the same idea as in *Butterworth Filters*, the descriptor of each output image is its uniform histogram with non-equidistant bins. Since seven bin histograms were used, the feature vectors contain seven components per filter.	$16 \times 7 \times 3 = 336$

Two bank datasets are selected to test the proposed methodology. There is a first bank which contains 105 images from VOPTICAL_I1 dataset [26], all of them taken over the same illumination conditions, which are considered to be the optimum ones by practitioners. This dataset contains the samples that are expected to be obtained in a real case situation and will be used to compute the performance of algorithms. It includes 29 open meshwork, 29 closed meshwork, 25 wave and 22 color fringe images. The second bank contains 406 images from VOPTICAL_Is dataset [27], taken over different illumination conditions. This bank will be used only to evaluate the sensitivity of algorithms to *noisy* data. It includes 159 open meshwork, 117 closed meshwork, 90 wave and 40 color fringe images.

The performance measures considered are the following:

- The *accuracy* is the percentage of correctly classified instances on a dataset with optimum illumination.
- The *robustness* is the classification accuracy in a noisy dataset, i.e., its accuracy when the images in the dataset show illuminations outside the optimum range. This measure is related to the generalization ability of the method when handling noisy inputs. Notice that the higher the robustness, the higher the generalization performance.
- The *feature extraction time* is the time that the texture analysis methods take to extract the selected features of a single image. Notice that this does not include

the training time of the classifier, which is not relevant for practical applications because the classifier will be trained offline. This also applies to FS, which is a preprocessing step that is performed offline.

The experimental procedure is detailed as follows,

1. Apply the five texture analysis methods (see Table 5.10) to the two banks of images. Moreover, the concatenation of all the features extracted by these five methods is also considered. As a result, six datasets with optimum illumination (105 images) and six datasets with different illuminations (406 images) are available. Notice that the feature extraction method chosen determines the number of features of the datasets, as can be seen in the first column of Table 5.11.
2. Apply three feature subset selection methods (CFS, consistency-based and INTERACT; see Chapter 2) to the datasets with optimum illumination to provide the subset of features that describe properly the given problem.
3. Train a SVM classifier (see Appendix A) with radial basis kernel and automatic parameter estimation. A 10-fold cross validation is used, so the average error across all 10 trials is computed.
4. Evaluate the effectiveness of feature selection in terms of three performance measures (accuracy, robustness to noise and feature extraction time), by means of the multiple-criteria decision-making method TOPSIS (see Appendix A, Section A.7.1).

The results obtained with and without feature selection are compared in terms of the three performance measures described above. Bear in mind that the column *None* in the tables shows the results when no feature selection was performed, and *Concatenation* stands for the concatenation of all the features of the five texture analysis methods. Note that the experimentation was performed on an Intel Core i5 CPU 760 at 2.80 GHz with 4 GB RAM.

The number of features selected by each of the three feature selection filters is summarized in Table 5.11. On average, CFS, Consistency-based filter (*Cons*) and INTERACT (*INT*) retain only the 4.9%, 1.6% and 3.2% of the features, respectively.

Table 5.11: Number of features

Texture analysis	Feature selection filter			
	None	CFS	Cons	INT
Butterworth filters	432	26	6	14
Discrete wavelet transform	36	10	8	7
Co-occurrence features	588	27	6	21
Markov random fields	660	24	13	15
Gabor filters	336	29	7	18
Concatenation	2052	56	6	29

5.2.1 Classification Accuracy

Table 5.12 shows the test accuracies for all pairwise texture analysis and feature selection methods after applying the SVM classifier over the VOPTICAL_I1 dataset. The best result for each is marked in boldface. As can be seen, all texture analysis techniques perform quite well, providing results over 84% accuracy. The best result is generated by the concatenation of all methods. Individually, Gabor filters and co-occurrence features without feature selection outperform the other methods. In fact, feature selection outperforms primal results in three out of six methods (Butterworth filters, the discrete wavelet transform and Markov random fields), while accuracy is almost maintained in co-occurrence features and the concatenation of all methods when CFS is applied. In conclusion, the best result is obtained by using the concatenation of all methods when no feature selection is performed (97.14%). Closely, the discrete wavelet transform with the INTERACT filter and the concatenation of all methods with CFS obtain an accuracy of 96.19%. Notice that although these results improve the previous slightly in accuracy [18] (95%), the goal of the work presented herein is to reduce the processing time whilst maintaining accuracy in order to be able to use the system in the daily clinical routine.

Table 5.12: Mean test classification accuracy (%), VOPTICAL_I1 dataset

Texture analysis	Feature selection filter			
	None	CFS	Cons	INT
Butterworth filters	91.42	**93.33**	83.81	86.67
Discrete wavelet transform	88.57	91.43	94.29	**96.19**
Co-occurrence features	**95.24**	94.29	86.67	93.33
Markov random fields	84.76	**85.71**	83.81	75.24
Gabor filters	**95.24**	91.43	86.67	86.67
Concatenation	**97.14**	96.19	87.62	93.33

5.2.2 Robustness to Noise

In some cases, data are taken over different illumination conditions which are not optimal. For this reason, it is also necessary to evaluate the sensitivity of the proposed methodology to noisy data. Table 5.13 shows the robustness of the six different methods on the VOPTICAL_IS dataset. Co-occurrence features and the concatenation of all methods obtain remarkably better results than the remaining methods. Both methods obtain values of robustness over 90% for some configurations. In particular, the best result is obtained by using the concatenation of all methods when the

CFS filter is used (93.84%). In relative terms, co-occurrence features and the concatenation of all methods deteriorate their mean classification accuracy by 2.66% and 4.59%, respectively (mean differences between the values contained in Tables 5.12 and 5.13).

Table 5.13: Robustness: Mean test accuracy (%), VOPTICAL_IS dataset

Texture analysis	Feature selection filter			
	None	CFS	Cons	INT
Butterworth filters	**88.18**	84.98	71.92	79.56
Discrete wavelet transform	82.27	**88.18**	86.21	86.70
Co-occurrence features	92.17	**92.36**	85.22	89.16
Markov random fields	**83.99**	76.35	70.94	70.69
Gabor filters	**89.90**	85.22	69.46	82.51
Concatenation	92.61	**93.84**	77.59	91.87

However, the remaining methods deteriorate their mean classification accuracy by between 6.78% and 8.23%. Note also that the illumination levels affect the robustness to different degrees. The brighter the illumination, the lower the robustness to noise. This also happens to practitioners when performing this task manually. For this reason, their experience in controlling the illumination level during the acquisition stage is the cornerstone to ensuring good classification performance.

5.2.3 Feature Extraction Time

Tear film lipid layer classification is a real-time task, so the time a method takes to process an image should not be a bottleneck. After applying feature selection and therefore reducing the number of input attributes, the time needed for analyzing a single image with any of the six methods was also reduced, as can be seen in Table 5.14.

In general terms, Butterworth filters, the discrete wavelet transform and Gabor filters require a negligible processing time to extract the features of an image (regardless of whether or not feature selection is applied as a preprocessing step). Moreover, Markov random fields takes a potentially acceptable amount of time for practical applications, even when no feature selection is applied, although it could not work in real time. The co-occurrence features method has been known to be slow and, despite the authors implementing an optimization of the method [28], it presents unacceptable extraction time. Regarding the time of the concatenation of all methods, note that it is influenced by the time of co-occurrence features. Co-occurrence features and the concatenation of all methods are only acceptable for practical applications when consistency-based or INTERACT filters are used. Consistency-based

Table 5.14: Feature extraction time (s)

Texture analysis	Feature selection filter			
	None	CFS	Cons	INT
Butterworth filters	0.22	0.15	**0.04**	0.07
Discrete wavelet transform	0.03	**0.01**	**0.01**	**0.01**
Co-occurrence features	102.18	27.01	**0.05**	9.86
Markov random fields	13.83	0.50	**0.27**	0.31
Gabor filters	0.42	0.18	**0.06**	0.11
Concatenation	116.68	37.04	**0.05**	9.96

filter selects fewer features (see Table 5.11) and consequently the processing time when this filter is used is smaller. Co-occurrence features is the core behind the good performance of the concatenation of all methods. This is demonstrated by further experiments showing that the concatenation of the other four methods achieves a maximum accuracy of 93.33% and robustness of 88.91%. These results are significantly worse (around 4%) than the best results obtained by the concatenation of all methods.

5.2.4 *Overall Analysis*

In general terms, we can assert that in a field with a very large number of features, feature selection filters play a significant role in reducing the cost of obtaining data and the complexity of the classifier. The consistency-based filter performed the most aggressive selection, retaining only the 1.6% of the features (see Table 5.11). CFS retained three times more features (4.9%) than the former. Halfway, INTERACT selected in average 3.2% of features. Moreover, in most cases the test accuracy is improved or maintained with a remarkable reduction in the number of features when feature selection is used (see Table 5.12). The effectiveness of feature selection on tear film lipid layer classification was then demonstrated, paving the way for its use in daily clinical routine.

Evaluating the performance of the methods for texture analysis presented herein is a multiobjective problem defined in terms of accuracy, robustness, and feature extraction time. Butterworth filters, the discrete wavelet transform and Gabor filters obtain competitive classification accuracies in short spans of time (see Tables 5.12 and 5.14). However, these methods are very sensitive to noisy data (see Table 5.13) which make them inappropriate for practical applications. On the other hand, the co-occurrence features method presents competitive results in classification accuracy and generalization (see Tables 5.12 and 5.14). However, the time the method takes to extract its features is an impediment (see Table 5.14). The concatenation of all methods improves the previous results but at the expense of an even longer feature extraction time.

Table 5.15 shows the TOPSIS values obtained for every method when the weights of each criteria are set equally. Note that the larger the value, the better the method. The top three methods are marked in bold. As can be seen, those methods with the best balance among classification accuracy, robustness to noise and feature extraction time are ranked in higher positions. In particular, Gabor filters with no feature selection, the discrete wavelet transform with INTERACT filter, and the concatenation of all methods with INTERACT filter rank in the top three positions of the ranking. However, those methods with good performance in accuracy and robustness but very long feature extraction time are penalized; e.g., co-occurrence features or the concatenation of all methods, with no feature selection in both cases.

Table 5.15: TOPSIS values obtained for every method when $w = [1/3, 1/3, 1/3]$

Texture analysis	Feature selection filter			
	None	CFS	Cons	INT
Butterworth filters	0.9774	0.9773	0.8159	0.9008
Discrete wavelet transform	0.9344	0.9775	0.9848	**0.9900**
Co-occurrence features	0.3431	0.9416	0.9281	0.9812
Markov random fields	0.8670	0.8691	0.8081	0.6923
Gabor filters	**0.9954**	0.9686	0.8295	0.9164
Concatenation	0.3066	0.8986	0.8988	**0.9853**

A more detailed look at the results contained in Tables 5.12, 5.13 and 5.14 reveals that the combination of all methods in both CFS filtering configuration and without feature selection obtain the best results in terms of accuracy and robustness. These two configurations are in the Pareto front [29] of accuracy versus robustness (see Figure 5.4). In multiobjective optimization, the Pareto front is defined as the border between the region of feasible points (not strictly dominated by any other), for which all constraints are satisfied, and the region of unfeasible points (dominated by others). In this case, solutions are constrained to maximize accuracy and robustness.

The suitability of these two solutions to the problem in question is also corroborated by TOPSIS. Table 5.16 shows the TOPSIS values when only accuracy and robustness are considered (note that the third term in the weight vector is considered to be the feature extraction time). The concatenation of all methods without feature selection and with CFS filtering rank first and second, respectively.

However, the time for extracting the features must be shortened for practical applications. Thus, a case study in which a deeper analysis for feature selection is carried out is presented in the next section. Note that the number of features in the concatenation of all methods without feature selection is too large (2,052 features) to be optimized by hand. Therefore, we will focus on the concatenation of all methods with CFS (56 features).

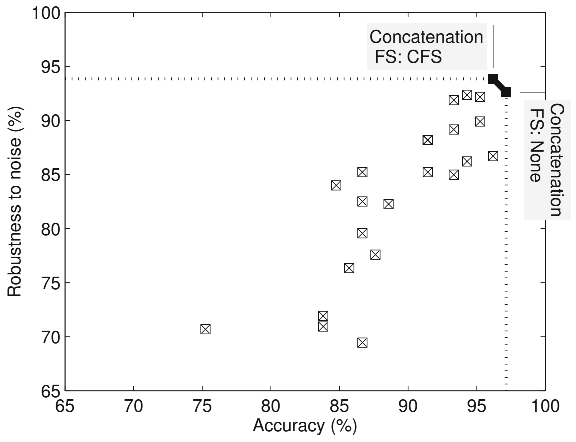


Fig. 5.4: Pareto front of a multiobjective optimization problem based on accuracy and robustness to noise

Table 5.16: TOPSIS values obtained for every method when $w = [1/2, 1/2, 0]$

Texture analysis	Feature selection filter			
	None	CFS	Cons	INT
Butterworth filters	0.8983	0.9017	0.1694	0.4722
Discrete wavelet transform	0.6567	0.8986	0.9377	0.9629
Co-occurrence features	**0.9923**	0.9846	0.6233	0.9539
Markov random fields	0.4777	0.3361	0.1601	0.0009
Gabor filters	0.9829	0.8526	0.2717	0.5526
Concatenation	**0.9991**	**0.9987**	0.4769	0.9706

5.2.5 The Concatenation of All Methods with CFS: A Case Study

When using feature selection, features are selected according to some specific criteria depending on the method. Filters remove features based on redundancy and relevance, but they do not take into account the costs for obtaining them. Note that the cost for obtaining a feature depends on the procedures required to extract it. Therefore, each feature has an associated cost that can be related to financial cost, physical risk or computation demands. This is the case of co-occurrence features and, consequently, the concatenation of all methods. In co-occurrence features the cost for obtaining the 588 features is not homogeneous. Features are vectorized in groups of 28 related to distances and channels in the color space. Each group of 28 features corresponds with the mean and range of 14 statistics across the co-occurrence matrices.

If we focus on co-occurrence features when using CFS, the number of features was reduced by 95.41% (from 588 to 27) but the processing time was not reduced in the same proportion, being 27.01 instead of the initial 102.18 seconds (a reduction of 73.57%). This fact clearly shows that computing some of the 588 features takes longer than computing others. Some experimentation was performed on the time the method takes to compute each of the 14 statistics. Results disclosed that computing the fourteenth statistic, which corresponds with the maximal correlation coefficient [21], takes around 96% of the total time. So the time for obtaining a single matrix is negligible compared to the time for computing the fourteenth statistic. Therefore, the key to reducing the feature extraction time is to reduce the number of fourteenth statistics in the selection.

Table 5.17: Co-occurrence features selected by CFS over the concatenation of all methods, in which features corresponding with the fourteenth statistic are marked in bold

Distance	Component in the color space		
	L	*a*	*b*
1	–	29,50	66
2	**98**	121,133	–
3	193	–	230
4	267,268,275,276,277	–	321
5	**350**,359	–	–
6	**434**,443,446	–	492,502
7	**518**	**546**	576

In the case of the concatenation of all methods with CFS, the filter selects 56 features (see Table 5.11) distributed as follows: 17 features of Butterworth filters, one of the discrete wavelet transform, 24 of co-occurrence features, one of Markov random fields, and 13 of Gabor filters. Five of the features selected in co-occurrence features correspond with the fourteenth statistic (see Table 5.17). In co-occurrence features, the cost of obtaining the statistics also depends on the distance and component in the color space. On the one hand, the longer the distance the larger the number of matrices to compute (and so, the higher the processing time). On the other hand, the differences of color have little contrast, so the colorimetric components of the Lab color space are minimal. As a consequence, the matrices within components *a* and *b* have smaller dimension than the matrices within component *L*. As expected, the smaller the dimension, the shorter the time to compute a statistic.

Computing the five fourteenth statistics in the different distances and components take: 3.12 s (feature 98), 8.23 s (feature 350), 9.61 s (feature 434), 11.49 s (feature 518), and 4.81 s (feature 546). As can be seen, avoiding computing some of them will entail saving a significant amount of time. The aim here is to explore the impact of removing some of the five fourteenth statistics selected by CFS in terms of accuracy, robustness and time. There are five features within the fourteenth statistic so only $2^5 = 32$ different configurations need to be explored. An empirical evaluation of brute force is acceptable. Table 5.18 shows the performance of the different configurations in terms of accuracy, robustness and time. Each configuration corresponds with those features selected by CFS removing some fourteenth statistics. For purposes of simplicity, only the acceptable results are shown. It is assumed that one solution is unacceptable if it obtains a lower accuracy and robustness in a longer time span than another.

Table 5.18: Performance measures for the concatenation of all methods with CFS when some of the five fourteenth statistics are not selected. The best results are marked in bold

Features removed	Acc (%)	Rob (%)	Time (s)
{}, *baseline performance*	96.19	93.84	37.04
{98, 434}	**97.14**	**94.09**	24.31
{98, 434, 546}	**97.14**	93.84	19.83
{98, 350, 518, 546}	**97.14**	93.60	9.72
{98, 434, 518, 546}	**97.14**	92.86	8.34
{98, 350, 434, 518, 546}	**97.14**	92.61	**0.11**

In terms of accuracy and robustness to noisy data, the best result is obtained when removing the features {98, 434} (results of 97.14% and 94.09%, respectively), but at the expense of a quite long processing time (24.31). Note that this result even improves the baseline performance. In the remaining results, the classification accuracy is maintained whilst the feature extraction time is reduced, at the expense of only a slight deterioration in terms of robustness to noise (less than 2%).

It is also important to mention the effectiveness of CFS filter for selecting the most appropriate features. If we do not apply feature selection and we simply remove the fourteenth statistics from the 588 features corresponding with co-occurrence features in the concatenation of all methods, the accuracy and the robustness are 92.86% for both of them. That is, the accuracy is worse than the results shown in Table 5.18 and the robustness is not significantly different. As expected, the time is also longer: 14.74 s.

To sum up, the manual process done by experts could be automatized with the benefits of being faster and unaffected by subjective factors, with maximum accuracy over 97% and processing time under one second. The clinical significance of these results should be highlighted, as the agreement between subjective observers is between 91% and 100%.

5.3 Cost-Based Feature Selection

The most common approach followed by feature selection methods is to find either a subset of features that maximizes a given metric or an ordered ranking of the features based on this metric. However, there are some situations (such as the one presented in the previous section) where a user is interested not only in maximizing the merit of a subset of features, but also in reducing costs that may be associated to features. For example, for medical diagnosis, symptoms observed with the naked eye are costless, but each diagnostic value extracted by a clinical test is associated with its own cost and risk. In other fields, such as image analysis, the computational expense of features refers to the time and space complexities of the feature acquisition process [30]. This is a critical issue, specifically in real-time applications, where the computational time required to deal with one feature or another is crucial, and in the medical domain, where it is important to save economic costs and also to improve the comfort of a patient by preventing risky or unpleasant clinical tests (variables that can be also treated as costs).

Our goal is to obtain a trade-off between a filter metric and the cost associated with the selected features, in order to select relevant features with low cost. A general framework to be applied together with the filter approach is introduced. In this manner, any filter metric can be modified to take into account the cost associated with the input features. In this section, we will present an implementation of the framework based on ReliefF, to be applied to tear film lipid layer classification, presented in the previous section.

5.3.1 Description of the Method

The general idea consists of adding a term to the evaluation function of the filter to take into account the cost of the features. Since, to our best knowledge, all filters use an evaluation function, this evaluation function could be modified to contemplate costs in the following manner. Let M_S be the merit of the set of k features S, that is, the value originally returned by the function

$$M_S = EvF(S), \tag{5.1}$$

where EvF is the evaluation function. Let C_S be the average cost of S

$$C_S = \frac{\sum_{i=1}^{k} C_i}{k}, \tag{5.2}$$

where C_i is the cost of feature i. The evaluation function can be modified to become:

$$MC_S = M_S - \lambda C_S, \tag{5.3}$$

where λ is a parameter introduced in order to weight the influence of the cost in the evaluation. If λ is 0, the cost is ignored and the method works as the regular feature selection method that is being modified. If the merit of the set of features M_S is bounded between 0 and 1, and λ is between 0 and 1, the influence of the cost is less than the other term. If $\lambda = 1$, both terms have the same influence, and if $\lambda > 1$, the influence of the cost is greater than the influence of the other term. Notice that when using a ranker, which selects or evaluates one feature at a time, the cardinality of S is 1, and then C_S in (5.2) results in the cost of that single feature.

Note that the parameter λ needs to be left as a free parameter because determining the importance of the cost is highly dependent on the domain. For example, in a medical diagnosis, the accuracy cannot been sacrificed in favor of reducing economical costs. On the contrary, in some real-time applications, a slight decrease in classification accuracy is allowed in order to reduce the processing time significantly.

5.3.1.1 Minimum Cost ReliefF, mC-ReliefF

ReliefF is a multivariate ranker filter algorithm. It randomly selects an instance R_i, but then searches for k of its nearest neighbors from the same class c, nearest hits H_j, and also k nearest neighbors from each one of the different classes, nearest misses $M_j(c)$. It updates the quality estimation $W[A]$ for all attributes A depending on their values for R_i, hits H_j and misses $M_j(c)$. If instances R_i and H_j have different values of the attribute A, then this attribute separates instances of the same class, which clearly is not desirable, and thus the quality estimation $W[A]$ has to be decreased. On the contrary, if instances R_i and M_j have different values of the attribute A for a class then the attribute A separates two instances with different class values, which is desirable, so the quality estimation $W[A]$ is increased. Since ReliefF considers multiclass problems, the contribution of all the hits and all the misses is averaged. Besides, the contribution for each class of the misses is weighted with the prior probability of that class $P(c)$ (estimated from the training set). The whole process is repeated m times (where m is a user-defined parameter) and can be seen in Algorithm 5.1.

The function $diff(A,I_1,I_2)$ calculates the difference between the values of the attribute A for two instances, I_1 and I_2. If the attributes are nominal, it is defined as:

$$diff(A,I_1,I_2) = \begin{cases} 0; & value(A,I_1) = value(A,I_2) \\ 1; & otherwise \end{cases}$$

The modification of ReliefF we propose in this research, mC-ReliefF, consists of adding a term to the quality estimation $W[f]$ to take into account the cost of the features, as can be seen in (5.4).

Algorithm 5.1: Pseudocode of ReliefF algorithm

Data: training set D, iterations m, attributes a
Result: the vector W of estimations of the qualities of attributes

```
1 set all weights $W[A] := 0$
2 for $i \leftarrow 1$ to $m$ do
3     randomly select an instance $R_i$
4     find $k$ nearest hits $H_j$
5     for each class $c \neq class(R_i)$ do
6         from class $c$ find $k$ nearest misses $M_j(c)$
      end
  end
7 for $f \leftarrow 1$ to $a$ do
8     $W[f] := W[f] - \frac{\sum_{j=1}^{k} diff(f,R_i,H_j)}{(m \cdot k)} + \frac{\sum_{c \neq class(R_i)} \left[\frac{P(c)}{1-P(class(R_i))} \sum_{j=1}^{k} diff(f,R_i,M_j(c))\right]}{(m \cdot k)}$
  end
```

$$W[f] := W[f] - \frac{\sum_{j=1}^{k} diff(f,R_i,H_j)}{(m \cdot k)} + \frac{\sum_{c \neq class(R_i)} \left[\frac{P(c)}{1-P(class(R_i))} \sum_{j=1}^{k} diff(f,R_i,M_j(c))\right]}{(m \cdot k)} - \lambda \cdot C_f, \tag{5.4}$$

where C_f is the cost of the feature f, and λ is a free parameter introduced to weight the influence of the cost in the quality estimation of the attributes. When $\lambda > 0$, the greater the λ the greater the influence of the cost.

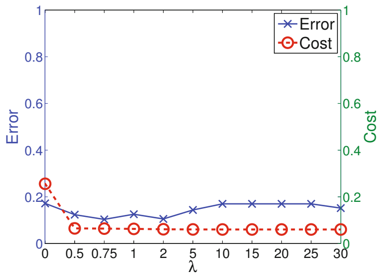
5.3.2 Experimental Results

The adequacy of the proposed framework using ReliefF is tested on the real problem of tear film lipid layer classification using the dataset VOPTICAL_I1 [26]. As mentioned in the previous section, this dataset consists of 105 images (samples) belonging to the four Guillon categories (classes). The methodology for TFLL classification proposed by [18] consists of extracting the *region of interest* (ROI) of an input image, and analyzing it based on color and texture information. Thus, the ROI in the RGB color space is transformed to the Lab color space and the texture of its three components of color (L, a and b) is analyzed. For texture analysis, the co-occurrence features method generates a set of *gray level co-occurrence matrices* (GLCM) for a specific distance and extracts 14 statistical measures from their elements. Then, the mean and the range of these 14 statistical measures are calculated across matrices, and so a set of 28 features is obtained. Distances from 1 to 7 in the co-occurrence features method and the three components of the Lab color space are considered, so the size of the final descriptor obtained from an input image is: 28 features $\times$ 7 distances $\times$ 3 components = 588 features. Notice that the cost for obtaining these features is not homogeneous. Features are vectorized in groups of 28 related to distances and components in the color space, where the higher the distance, the higher the cost. Plus, each group of 28 features corresponds with the mean and range of the 14 statistical measures calculated across the GLCMs. Among these statistical measures, it was shown that computing the so-called fourteenth statistic takes around 75% of the total time (see Section 5.2.5). Therefore, we have to deal with a dataset with a very variable cost (in this case, computational time) associated with the input features.

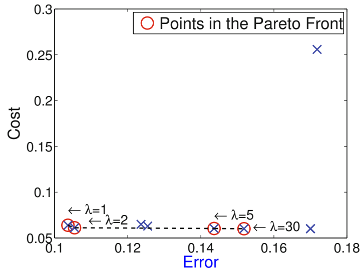
Figure 5.5 (top) shows the average error and cost after performing a ten-fold cross-validation for the VOPTICAL_I1 dataset for different values of λ, for three different sets of features. As expected, when λ increases, the cost decreases and the error either rises or is maintained. Regarding the different subsets of features, the larger the number of features, the higher the cost. The Kruskal-Wallis statistical test run on the results demonstrated that there are no significant differences among the errors achieved using different values of λ, whilst using a $\lambda > 0$ decreases the cost significantly. This situation happens when retaining 25, 35 and 50 features.

To shed light on the issue of which value of λ is better for the problem at hand, the Pareto front [29] for each alternative is showed in Figure 5.5 (bottom). In multi-objective optimization, the Pareto front is defined as the border between the region of feasible points, for which all constraints are satisfied, and the region of infeasible points. In this case, solutions are constrained to minimize classification error and cost. In Figure 5.5 (bottom), points (values of λ) in the Pareto front are marked with a red circle. All those points are equally satisfying the constraints, and it is the users' decision to minimize the cost or the classification error. On the other hand, choosing a value of λ outside the Pareto front would imply choosing a worse solution than any in the Pareto front.

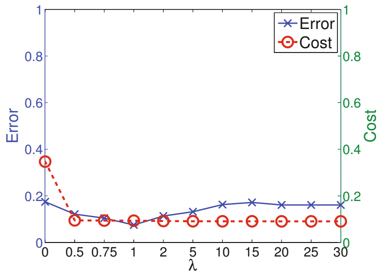
Table 5.19 reports the classification error and cost (in the form of time) for all the Pareto front points. Notice that as a 10-fold cross-validation was performed, the final subset of selected features is the union of the features selected in each fold, and that is why the number of features in the fifth column differs from the one in the first column. Even so, the reduction in the number of features is considerable. As expected, the higher the λ, the higher the error and the lower the time. The best result in terms of classification error was obtained with $\lambda = 0$ when retaining 50 features per fold. In turn, the lowest time was obtained with $\lambda = 30$ when retaining 25 features per fold, but at the expense of increasing the error in 8.54%. In this situation, the authors think that it is better to choose a trade-off between cost and error. The error obtained with $\lambda = 1$ when retaining 35 features is 7.55%, which is slightly higher than the best one, but no significant differences were found between them. With this combination the time required is 306.53 ms, which, although not the lowest time, is still under one second. The time required by previous approaches which deal with TFLL classification prevented their clinical use because they could not work in real time, since extracting the whole set of features took 38 seconds. Thus, since this is a real-time scenario where reducing the computing time is a crucial issue, having a processing time under one second leads to a significant improvement.



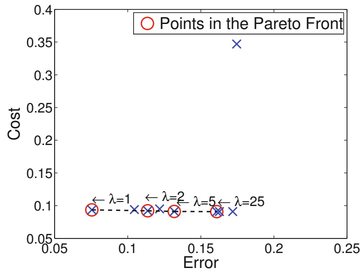
(a) Error/cost 25 feats



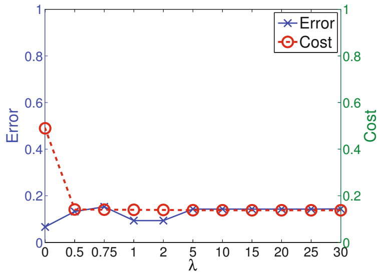
(b) Pareto front 25 feats



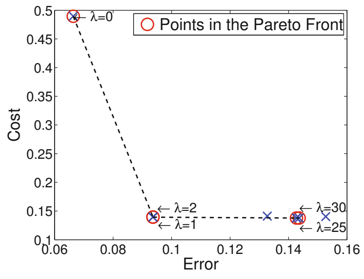
(c) Error/cost 35 feats



(d) Pareto front 35 feats



(e) Error/cost 50 feats



(f) Pareto front 50 feats

Fig. 5.5: Error/cost plots (left) and Pareto front (right) of VOPTICAL_I1 dataset for different values of λ, and different number of selected features (25, 35 and 50)

In this manner, the methodology for tear film lipid layer classification could be used in the clinical routine as a support tool to diagnose Evaporative Dry Eye.

Table 5.19: Mean classification error (%), time (ms), and number of features in the union of the 10 folds for the Pareto front points. Best error and time are marked in boldface

Feats	λ	Error	Time	Feats union
25	0.75	10.36	208.68	30
	2	10.55	206.46	30
	5	14.36	197.22	29
	30	15.18	**174.35**	26
35	1	7.55	306.53	43
	2	11.36	328.24	46
	5	13.18	273.11	39
	25	16.09	249.92	36
50	0	**6.64**	1377.04	82
	1	9.36	397.70	55
	2	9.36	412.14	57
	25	14.27	364.45	51
	30	14.36	364.45	51

5.4 Summary

Feature selection plays a crucial role in many real applications, since it reduces the number of input features and, most of the time, improves performance. This chapter presented two real problems where feature selection has proved to be useful for achieving better performance results.

The first problem was the classification of the KDD Cup 99 dataset, a benchmark in the intrusion detection field. A method based on the combination of discretization, filtering and classification algorithms was proposed, with the aim of maintaining the performance results of the classifiers but using a reduced set of features. The proposed method was able to improve the results obtained by the KDD Cup 99 Competition winner and also by other authors, with a significant reduction in the number of features used in two different approaches. The simplest one considers a binary classification and the second one a multiple class classification.

The second real application considered was related with tear film lipid layer classification. The time required by existing approaches dealing with this issue prevented their clinical use because they could not work in real time. In this chapter, a methodology for improving this classification problem was proposed, which includes the application of feature subset selection methods: CFS, Consistency-based, and INTERACT. Results obtained with this methodology surpass previous results in terms of processing time whilst maintaining accuracy and robustness to noise. In clinical terms, the manual process done by experts can be automated with the benefits of being faster and unaffected by subjective factors, with maximum accuracy over 97% and processing time under 1 s. The clinical significance of these results should be highlighted, as the agreement among subjective observers is between 91% and 100%.

Finally, we have proposed a general framework for cost-based feature selection, applicable to situations in which it is important to reduce the cost associated with the features, as is the case in tear film lipid layer classification. Results after performing classification with an SVM displayed that the approach is sound and allows the user to reduce the cost without compromising the classification error significantly, which can be very useful in fields such as medical diagnosis or real-time applications.

References

1. KDD Cup 99 Dataset. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html [Last access: November 2014].
2. Mukkamala, S., Sung, A. H. and Abraham, A. Intrusion detection using an ensemble of intelligent paradigms. *Journal of Network and Computer Applications*, 28(2):167–182, 2005.
3. Yang, Y. and Webb, G. I. A comparative study of discretization methods for naive-Bayes classifiers. In *Pacific Rim Knowledge Acquisition Workshop*, pages 159–173, 2002.
4. Fayyad, U. and Irani, K. Multi-interval discretization of continuous-valued attributes for classification learning. In *13th International Joint Conference on Artificial Intelligence*, pages 1022–1029, 1993.
5. Yang, Y. and Webb, G. I. Proportional k-interval discretization for naive-bayes classifiers. In *Machine Learning: ECML 2001*, pages 564–575. Springer, 2001.
6. Bolón-Canedo, V., Sánchez-Maroño, N. and Alonso-Betanzos, A. A combination of discretization and filter methods for improving classification performance in KDD Cup 99 dataset. In *IEEE International Joint Conference on Neural Networks*, pages 359–366, 2009.
7. Axelsson, S. The base-rate fallacy and its implications for the difficulty of intrusion detection. In *6th ACM Conference on Computer and Communications Security*, pages 1–7, 1999.
8. Grkabczewski, K. and Jankowski, N. Transformations of symbolic data for continuous data oriented models. In *Artificial Neural Networks and Neural Information Processing - ICANN/ICONIP 2003*, pages 359–366. Springer, 2003.
9. Forman, G. An extensive empirical study of feature selection metrics for text classification. *The Journal of Machine Learning Research*, 3:1289–1305, 2003.
10. Khoshgoftaar, T. M., Gao, K. and Ibrahim, N. H. Evaluating indirect and direct classification techniques for network intrusion detection. *Intelligent Data Analysis*, 9(3):309–326, 2005.
11. Allwein, E. L., Schapire, R. E. and Singer, Y. Reducing multiclass to binary: A unifying approach for margin classifiers. *The Journal of Machine Learning Research*, 1:113–141, 2001.
12. Elkan, C. Results of the KDD'99 classifier learning. *ACM SIGKDD Explorations Newsletter*, 1(2):63–64, 2000.
13. Fugate, M. and Gattiker, J. R. Computer intrusion detection with classification and anomaly detection, using SVMs. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(03):441–458, 2003.
14. Alonso-Betanzos, A., Sánchez-Maroño, N., Carballal-Fortes, F. M., Suárez-Romero, J. A. and Pérez-Sánchez, B. Classification of computer intrusions using functional networks. A comparative study. In *15th European Symposium on Artificial Neural Networks*, pages 25–27, 2007.
15. Mukkamala, S. and Sung, A. H. Feature ranking and selection for intrusion detection systems using Support Vector Machines. In *Second Digital Forensic Research Workshop*, 2002.
16. Guillon, J. P. Non-invasive Tearscope Plus routine for contact lens fitting. *Contact Lens and Anterior Eye*, 21:S31–S40, 1998.

17. García-Resúa, C., Giráldez Fernández, M. J., González Penedo, M. F., Calvo, D., Penas, M. and Yebra-Pimentel, E. New software application for clarifying tear film lipid layer patterns. *Cornea*, 32(4):538–546, 2013.
18. Remeseiro, B., Ramos, L., Penas, M., Martinez, E., G. Penedo, M. and Mosquera. A. Colour texture analysis for classifying the tear film lipid layer: A comparative study. In *IEEE International Conference on Digital Image Computing Techniques and Applications*, pages 268–273, 2011.
19. Gonzalez, R. and Woods, R. *Digital Image Processing*. Pearson/Prentice Hall, 2008.
20. Mallat, S. G. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.
21. Haralick, R. M., Shanmugam, K. and Dinstein, I. H. Textural features for image classification. *IEEE Transactions on Systems, Man and Cybernetics*, (6):610–621, 1973.
22. Woods, J. Two-dimensional discrete Markovian fields. *IEEE Transactions on Information Theory*, 18(2):232–240, 1972.
23. Gabor, D. Theory of communication. Part 1: The analysis of information. *Journal of the Institution of Electrical Engineers - Part III: Radio and Communication Engineering*, 93(26):429–441, 1946.
24. Ramos, L., Penas, M., Remeseiro, B., Mosquera, A., Barreira, N. and Yebra-Pimentel, E. Texture and color analysis for the automatic classification of the eye lipid layer. In *Advances in Computational Intelligence*, pages 66–73. Springer, 2011.
25. Çesmeli, E. and Wang, D. Texture segmentation using Gaussian-Markov random fields and neural oscillator networks. *IEEE Transactions on Neural Networks*, 12(2):394–404, 2001.
26. VOPTICAL_I1. VARPA optical dataset annotated by optometrists from the Faculty of Optics and Optometry, University of Santiago de Compostela (Spain) http://www.varpa.es/voptical_I1.html
27. VOPTICAL_Is. VARPA optical dataset annotated by optometrists from the Faculty of Optics and Optometry, University of Santiago de Compostela (Spain) http://www.varpa.es/voptical_Is.html,
28. Clausi, D. A. and Jernigan, M. E. A fast method to determine co-occurrence texture features. *IEEE Transactions on Geoscience and Remote Sensing*, 36(1):298–300, 1998.
29. Teich, J. Pareto-front exploration with uncertain objectives. In *Evolutionary Multi-Criterion Optimization*, pages 314–328. Springer, 2001.
30. Feddema, J. T., Lee, C. S. G. and Mitchell, O. R. Weighted selection of image features for resolved rate visual feedback control. *IEEE Transactions on Robotics and Automation*, 7(1):31–47, 1991.

Chapter 6
Emerging Challenges

Abstract This chapter reveals the new challenges that the researchers in feature selection need to face as a consequence of the explosion of "Big Data" and, in particular, "Big Dimensionality." Section 6.1 states the implications of having to deal with an unprecedented number of features. Then, Section 6.2 defines the concept of *scalability* in the current scenario. We also mention new approaches that need to be studied nowadays by feature selection researchers, such as distributed feature selection (Section 6.3) and real-time processing (Section 6.4). Finally, Section 6.5 summarizes and discusses the contents of this chapter.

Continual advances in computer-based technologies have enabled researchers and engineers to collect data at an increasingly fast pace. To address this challenge, feature selection becomes an imperative preprocessing step which needs to be adapted and improved to handle high-dimensional data. So far, this book was devoted to studying feature selection methods and their adequacy for being applied to data of high dimensionality. However, there are still an important number of emerging challenges that researchers need to deal with. In this chapter, we will discuss the core challenges that lie ahead.

6.1 Millions of Dimensions

In the new era of Big Data, machine learning methods need to be adapted in order to be able to deal with this unprecedented scale of data. Analogously, the term Big Dimensionality is coined to refer to the unprecedented number of features that is scaling to levels which nowadays render existing state-of-the-art machine learning methods inadequate [1].

If we take a look at the widely used UC Irvine Machine Learning Repository [2], we can observe that in the 1980s the maximum dimensionality of the data was only about 100; in the 1990s this number increased to more than 1,500; and in 2009 it further increased to more than three million. If we perform the same analysis on the popular LIBSVM Database [3], we can observe that in the 1990s the maximum dimensionality of the data was about 62,000, which increased to more than 16 million in the 2000s and to more than 29 million in the 2010s. It is worth noting that seven of the datasets that appeared in the last nine years in these two repositories have dimensionality of the order of millions.

In this scenario, existing state-of-the-art feature selection methods are confronted with key challenges that can deteriorate their performance. As an example, Zhai et al. [1] pointed out that when state-of-the-art SVM-RFE and mRMR feature selectors had to deal with the psoriasis Single-Nucleotide Polymorphism (SNP) dataset, which is composed of *only* half a million features, it took more than a day of computational effort to crunch the data.

What is more, a great number of state-of-the-art feature selection methods are based on computing pairwise correlations in their algorithm designs. Imagine the implications of this when dealing with a million features; it is necessary to cope with a trillion correlations. Note that this poses an enormous challenge for machine learning researchers that has not been addressed yet.

6.2 Scalability

The great majority of existing learning algorithms were developed when dataset sizes were much smaller; nowadays distinct trade-offs are required for small-scale and large-scale learning problems. Small-scale learning problems are subject to the usual approximation-estimation trade-off. In the case of large-scale learning problems, the trade-off is more complex because it involves not only the accuracy but also the computational complexity of the learning algorithm. Moreover, the problem here is that the majority of algorithms were designed under the assumption that the dataset would be represented as a single memory-resident table. So if the entire dataset does not fit in main memory, these algorithms are useless.

For all these reasons, scaling up learning algorithms is a trending issue. The organization of the workshops "PASCAL Large Scale Learning Challenge" at the 25th International Conference on Machine learning (ICML 2008), and "Big Learning" at the conference of the Neural Information Processing Systems Foundation (NIPS 2011) is a case in point. Scaling up is desirable because increasing the size of the training set often increases the accuracy of algorithms [4]. For scaling up learning algorithms, the issue is not so much of speeding up a slow algorithm as of tuning an impracticable algorithm into a practical one. The crucial issue is seldom how fast you can run on a particular problem, but rather how large a problem you can deal with [5].

Scalability is defined as the effect that an increase in the size of the training set has on the computational performance of an algorithm: accuracy, training time and allocated memory. Thus the challenge is to find a balance among them or, in other words, get “good enough” solutions as “fast” as possible and as “efficiently” as possible. This issue becomes critical in situations in which temporal or spatial constraints exist, such as real-time applications dealing with large datasets, unapproachable computational problems requiring learning, or initial prototyping requiring quick-implemented solutions.

Feature selection methods can be helpful in scaling machine learning algorithms as they reduce the input dimensionality and therefore the runtime required by an algorithm. However, when dealing with a dataset which contains a huge number of features and samples, the scalability of a feature selection method also becomes of crucial importance. Since most of the existing feature selection techniques were designed to process small-scale data, their efficiency can be downgraded, if not totally inapplicable, with high-dimensional data.

In this scenario, feature selection researchers need to be focused not only on the accuracy of the selection but also on other aspects. Stability, that is, the sensitivity of the results to training set variations, is one such factor, with a few studies published regarding the behavior of filters in the case where the training set is small, but the number of features can be high [6, 7, 8]. The other important aspect, scalability, that is, the behavior of feature selection methods in the case where the training set is increasingly high, is even scarcer in the scientific literature [9]. The studies are mainly concentrated on obtaining scalability in a particular application [10], modifying certain previously existing approaches [11], or adopting online [12] or parallel [13] approaches. In general, one can say that most of the classical feature selection approaches that are univariate—that is, each feature is considered separately—have an important advantage in scalability, but at the cost of ignoring feature dependencies, and thus perhaps leading to weaker performances than other feature selection techniques. To improve performance, multivariate filter techniques are proposed, but at the cost of reducing scalability [14]. In this situation, the scalability of a feature selection method becomes extremely important and deserves more attention by the scientific community. One of the solutions commonly adopted to deal with the scalability issue is to distribute the data into several processors, which will be commented on in the following section.

6.3 Distributed Feature Selection

Traditionally, feature selection is applied in a centralized manner, i.e., a single learning model to solve a given problem. However, nowadays the data is sometimes distributed, and then feature selection may take advantage of processing multiple subsets in sequence or concurrently. There are several ways in which a feature selection task could be distributed [15]:

(a) If all the data is together in one very large dataset, we can distribute it on several processors, run an identical feature selection algorithm on each one and combine the results.

(b) The data may inherently be in different datasets on different locations, for example, in different parts of a company or even in different cooperating organizations. As with the previous case, we could run an identical feature selection on each one and combine the results.
(c) An extreme case of a large data volume is *streaming data* arriving in effectively a continuous infinite stream in real time. If the data is all coming to a single source, different parts of it could be processed by different processors acting in parallel. If it is coming into several different processors, it could be handled in a similar way to (b).
(d) An entirely different situation arises when we have a dataset that is not particularly large, but we wish to generate several or many different feature selection methods from it and then combine the results using some kind of voting system in order to learn unseen instances. In this case we might have the whole dataset on a single processor, accessed by different feature selection methods (possibly identical or possibly different) accessing all or part of the data. This approach is known as *ensemble learning* and has recently received a significant amount of attention [16].

As mentioned above, most existing feature selection methods are not expected to scale well when dealing with millions of features, and their efficiency may significantly deteriorate or they may even become inapplicable. Therefore, a possible solution might be to distribute the data, run a feature selection on each partition and then combine the results. There are two main techniques for partitioning and distributing data: vertically, i.e., by features, and horizontally, i.e., by samples. Distributed learning has been used to scale up datasets that are too large for batch learning in terms of samples [17, 18, 19]. While not common, there are some other developments that distribute the data by features [20, 21]. When dealing with datasets of Big Dimensionality, researchers need to focus on the latter approach: partitioning by features.

Among the existing paradigms for performing distributed learning, Apache Spark [22], which is a fast and general engine for large-scale data processing, has become very popular in the last few years. MLib [23] has been created with the aim of being a scalable machine learning library which contains algorithms developed within the Apache Spark paradigm. In fact, it already includes a number of learning algorithms such as SVM and naive Bayes classification, k-means clustering, etc. Nevertheless, feature selection algorithms have not been included in this library yet; thus this poses a key challenge to machine learning researchers, as well as a great opportunity to initiate a new line of research.

Another open line of research is the use of Graphics Processing Units (GPUs) to distribute and thus accelerate the calculations made in feature selection algorithms. Parallel algorithms running on GPUs can often achieve up to 100× speedup over similar CPU algorithms, with many existing applications for physics simulations, signal processing, financial modeling, neural networks, and countless other fields. The challenge now is to take advantage of GPU capabilities to adapt existing state-of-the-art feature selection methods to cope with millions of features in an effective yet accurate way.

6.4 Real-Time Processing

With the advent of Big Data, data is being collected at an unprecedented fast pace, and therefore it needs to be processed in a short time as well. In an era where the prevalence of social media networks and portable devices dominates our day-to-day life, it is necessary to develop sophisticated methods capable of dealing with big amounts of data in real time, such as spam detection or video or image detection [1].

To deal with data streams that flow continuously, classical batch learning algorithms cannot be applied and it is necessary to employ online approaches. Online learning consists of continuously revising and refining a model by incorporating new data as they arrive, and it has become a trending area in the last few years since it allows important problems such as concept drift or management of extremely high-dimensional datasets to be solved. For this reason, advances in this field have recently appeared. However, online feature selection has not evolved in line with online learning. Zhang et al. [24] proposed an incremental computation feature subset selection algorithm which, originating from Boolean matrix technique, selects useful features for the given data objective efficiently. Nevertheless, the efficiency of the feature selection method has not been tested with an incremental machine learning algorithm. In [25], the idea of a dynamic feature space was mentioned. The features that are selected based on an initial collection of training documents are the ones that are subsequently considered by the learner during the operation of the system. However, these features may vary over time and in some applications an initial training set is not available. Katakis et al. [25] applied incremental feature selection combined with what they called a feature-based learning algorithm to deal with online learning in high-dimensional data streams. This framework is applied to a special case of concept drift inherent in textual data streams, that is, the appearance of new predictive words over time. The problem with this approach is that they assume that features have discrete values. Perkins et al. [26] presented a novel and flexible approach, called grafting, which treats the selection of suitable features as an integral part of learning a predictor in a regularized learning framework. To make it suitable for large problems, grafting operates in an incremental iterative fashion, gradually building up a feature set while training a predictor model using gradient descent. Perkins and Theiler [27] tackle the problem in which, instead of all features being available from the start, features arrive one at a time. Online Feature Selection (OFS) assumes that, for any reason, it is not acceptable to wait until all features have arrived before learning begins; therefore one needs to derive a mapping function f from the inputs to the outputs that is as "good as possible" using a subset of only the features seen so far. In [28], the power of OFS in the image processing domain was demonstrated by applying it to the problem of edge detection. In a recent work [29], a promising alternative method, Online Streaming Feature Selection (OSFS), to select strongly relevant and nonredundant features online, is presented. Finally, some other research has been found in the literature comprising online feature selection and classification. In [30], the authors presented an online learning algorithm for feature extraction and classification, implemented for impact acoustic signals to sort hazelnut kernels. Levi and Ullman [31] proposed classifying images by ongoing feature selection. However, their approach only uses at each stage a small subset of the training data. In [32], online feature selection is performed based on the weights assigned to each input of the classifiers.

As can be seen, online feature selection has been faced mostly individually, i.e., by selecting features previously in a single step independently of the online machine learning step, or by performing online feature selection without performing online classification afterwards. Therefore, achieving real-time analysis and prediction on high-dimensional datasets is still an open challenge of computational intelligence on portable platforms.

6.5 Summary

As can be seen throughout this book, feature selection is a much needed preprocessing step when dealing with large-scale data. It is useful for coping with scenarios with a large number of both input features and samples. However, it is especially important now that the term "Big Dimensionality" has been introduced as a consequence of the explosion of "Big Data."

This chapter has revealed the new challenges that researchers need to face, since there is a lack of studies on the evolution of data dimensionality. In the last few years, datasets with a number of features in the order of millions have been produced, and there is evidence that this number will only continue to increase, given the rapid advancements in computing and information technologies. As pointed out by Zhai et al. [1], "a forecast of 40 billion features in dimensions is to be expected by year 2020."

With this new scenario, there are a number of opportunities open for machine learning researchers. The need for scalable yet efficient methods is obvious, since existing feature selection methods will be inadequate for coping with this unprecedented number of features. Moreover, the society has expressed new necessities, such as distributed learning or real-time processing, where there is still an important gap that needs to be filled.

In conclusion, this explosion in the number of features does not have to be regarded as the *curse of Big Dimensionality*, but as the *blessing of Big Dimensionality*, and thus researchers must embrace the new opportunities and challenges that have recently arisen.

References

1. Zhai, Y., Ong, Y. and Tsang, I. The emerging ”Big Dimensionality”. *IEEE Computational Intelligence Magazine*, 9(3):14–26, 2014.

2. K. Bache and M. Lichman. UCI Machine Learning Repository, 2013. University of California, Irvine, School of Information and Computer Sciences. http://archive.ics.uci.edu/ml
3. Chang, C. C. and Lin, C. J. LIBSVM: A library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.
4. Catlett, J. *Megainduction: Machine learning on Very Large Databases*. Ph.D. Thesis, University of Sydney, 1991.
5. Provost, F. and Kolluri, V. A survey of methods for scaling up inductive algorithms. *Data Mining and Knowledge Discovery*, 3(2):131–169, 1999.
6. Brown, G., Pocock, A., Zhao, M. J. and Luján, M. Conditional likelihood maximisation: A unifying framework for information theoretic feature selection. *The Journal of Machine Learning Research*, 13(1): 27–66. 2012.
7. Fahad, A., Tari, Z., Khalil, I., Habib, I. and Alnuweiri, H. Toward an efficient and scalable feature selection approach for Internet traffic classification. *Computer Networks*, 57:2040–2057, 2013.
8. Gulgezen, G., Cataltepe, Z. and Yu, L. Stable and accurate feature selection. In *Machine Learning and Knowledge Discovery in Databases*, pages 455–468. Springer, 2009.
9. Peteiro-Barral, D., Bolón-Canedo, V., Alonso-Betanzos, A., Guijarro-Berdiñas, B. and Sánchez-Maroño, N. Scalability analysis of filter-based methods for feature selection. *Advances in Smart Systems Research*, 2(1):21–26, 2012.
10. Luo, D., Wang, F., Sun, J., Markatou, M., Hu, J. and Ebadollahi, S. SOR: Scalable orthogonal regression for non-redundant feature selection and its healthcare applications. In *SIAM Data Mining Conference*, pages 576–587, 2012.
11. Sun, Y., Todorovic, S. and Goodison, S. A feature selection algorithm capable of handling extremely large data dimensionality. In *SIAM International Conference in Data Mining*, pages 530–540, 2008.
12. Hoi, S. C. H., Wang, J., Zhao, P. and Jin, R. Online feature selection for mining big data. In *1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, pages 93–100. ACM, 2012.
13. Zhao, Z., Zhang, R., Cox, J., Duling, D. and Sarle, W. Massively parallel feature selection: An approach based on variance preservation. *Machine Learning*, 92:195–220, 2013.
14. Alonso-Betanzos, A., Bolón-Canedo, V., Fernández-Francos, D., Porto-Díaz, I. and Sánchez-Maroño, N. Up-to-Date feature selection methods for scalable and efficient machine learning In *Igelnik, B., Zurada, J.M., eds., Efficiency and Scalability Methods for Computational Intellect*, pages 1–26. IGI Global, 2013.
15. Bramer, M. *Principles of Data Mining*. Springer, 2007.
16. Kuncheva, L. I. and Whitaker, C. J. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, 2003.
17. Chan, P. K. and Stolfo, S. J. Toward parallel and distributed learning by meta-learning. In *AAAI Workshop in Knowledge Discovery in Databases*, pages 227–240, 1993.
18. Ananthanarayana, V. S., Subramanian, D. K. and Murty, M. N. Scalable, distributed and dynamic mining of association rules. *High Performance Computing*, pages 559–566, 2000.
19. Tsoumakas, G. and Vlahavas, I. Distributed data mining of large classifier ensembles. In *2nd Hellenic Conference on Artificial Intelligence*, pages 249–256, 2002.
20. McConnell, S. and Skillicorn, D. B. Building predictors from vertically distributed data. In *Conference of the Centre for Advanced Studies on Collaborative Research*, pages 150–162. IBM Press, 2004.
21. Skillicorn, D. B. and McConnell, S. M. Distributed prediction from vertically partitioned data. *Journal of Parallel and Distributed Computing*, 68(1):16–36, 2008.
22. Apache Spark. https://spark.apache.org
23. MLib / Apache Spark. https://spark.apache.org/mllib
24. Zhang, C., Ruan, J. and Tan, Y. An incremental feature subset selection algorithm based on boolean matrix in decision system. *Convergence Information Technology*, pages 16–23, 2011.

25. Katakis, I., Tsoumakas, G. and Vlahavas, I. Dynamic feature space and incremental feature selection for the classification of textual data streams. *Knowledge Discovery from Data Streams*, pages 107–116, 2006.
26. Perkins, S., Lacker, K. and Theiler, J. Grafting: Fast, incremental feature selection by gradient descent in function space. *The Journal of Machine Learning Research*, 3:1333–1356, 2003.
27. Perkins, S. and Theiler, J. Online feature selection using grafting. In *International Conference on Machine Learning*, pages 592–599, 2003.
28. Glocer, K., Eads, D. and Theiler, J. Online feature selection for pixel classification. In *22nd International Conference on Machine Learning*, pages 249–256, 2005.
29. Wu, X., Yu, K., Wang, H. and Ding, W. Online streaming feature selection. In *27nd International Conference on Machine Learning*, pages 1159–1166, 2010.
30. Kalkan, H. and Çetisli, B. Online feature selection and classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2124–2127, 2011.
31. Levi, D. and Ullman, S. Learning to classify by ongoing feature selection. *Image and Vision Computing*, 28(4):715–723, 2010.
32. Carvalho, V. R. and Cohen, W. W. Single-pass online learning: Performance, voting schemes and online feature selection. In *12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 548–553, 2006.

Appendix A
Experimental Framework Used in This Book

This appendix describes the materials and methods used in this book, as well as the evaluation techniques and performance metrics employed.

A.1 Software Tools

The experiments performed in this book were executed using the software tools Matlab and Weka, which are described in the following paragraphs.

- Matlab [1] is a numerical computing environment, well known and widely used by scientific researchers. It was developed by MathWorks in 1984 and its name comes from *Matrix Laboratory*. Matlab allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran.
- Weka (Waikato Environment for Knowledge Analysis) [2] is a collection of machine learning algorithms for data mining tasks. The algorithms can be either applied directly to a dataset or called from your own Java code. Weka contains tools for data preprocessing, classification, regression, clustering, association rules, and visualization. It is also well suited for developing new machine learning schemes.

A.2 Datasets

A common procedure for testing the adequacy of a given feature selection methods is to check its performance over a benchmark dataset. To facilitate this task, several repositories of data exist that will be listed in Section A.2.1. In this book, specifically, we have used synthetic datasets (the relevant features are known a priori), classical datasets extracted from the widely used UCI repository [3] and also some datasets that belong to the category of DNA microarray datasets. After presenting the most popular repositories, subsequent subsections will present the main characteristics of the datasets employed in this book.

A.2.1 Data Repositories

In this subsection we present some public data repositories covering a wide spectrum of data which could be useful for feature selection researchers.

- Miscellaneous repositories:
  - *The UC Irvine Machine Learning Repository* (UCI), from University of California, Irvine :
    `http://archive.ics.uci.edu/ml/`
  - *UCI KDD Archive*, from University of California, Irvine:
    `http://kdd.ics.uci.edu`
  - *LIBSVM Database*:
    `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`
  - *Public Data Sets*, from Amazon Web Services:
    `http://aws.amazon.com/datasets`
  - *The Datahub*:
    `http://datahub.io/dataset`

- Specific repositories for microarray data:
  - *ArrayExpress*, from the European Bioinformatics Institute:
    `http://www.ebi.ac.uk/arrayexpress/`
  - *Cancer Program Data Sets*, from the Broad Institute:
    `http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi`
  - *Dataset Repository*, from the Bioinformatics Research Group of Universidad Pablo de Olavide:
    `http://www.upo.es/eps/bigs/datasets.html`
  - *Feature Selection Datasets*, from Arizona State University:
    `http://featureselection.asu.edu/datasets.php`
  - *Gene Expression Model Selector*, from Vanderbilt University:
    `http://www.gems-system.org`
  - *Gene Expression Omnibus*, from the National Insititutes of Health:
    `http://www.ncbi.nlm.nih.gov/geo/`
  - *Gene Expression Project*, from Princeton University:
    `http://genomics-pubs.princeton.edu/oncology/`
  - *Kent Ridge Bio-Medical Dataset Repository*, from the Agency for Science, Technology and Research:
    `http://datam.i2r.a-star.edu.sg/datasets/krbd`

– *Stanford Microarray Database*, from Stanford University:
  http://smd.stanford.edu/

A.2.2 Synthetic Datasets

Several authors choose to use artificial data since the desired output is known; therefore a feature selection algorithm can be evaluated independently of the classifier used. Although the final goal of a feature selection method is to test its effectiveness over a real dataset, the first step should be on synthetic data. The reason for this is twofold [4]:

1. Controlled experiments can be developed by systematically varying chosen experimental conditions, such as adding more irrelevant features or noise in the input. This fact facilitates drawing more useful conclusions and testing the strengths and weaknesses of the existing algorithms.
2. The main advantage of artificial scenarios is the knowledge of the set of optimal features that must be selected; thus the degree of closeness to any of these solutions can be assessed in a confident way.

The synthetic datasets used in this book try to cover different problems: increasing number of irrelevant features, redundancy, noise in the output, alteration of the inputs, nonlinearity of the data, etc. These factors complicate the task of the feature selection methods, which are very affected by them, as will be shown afterwards. Besides, some of the datasets have a significantly higher number of features than samples, which implies an added difficulty for the correct selection of the relevant features. Table A.1 shows a summary of the main problems covered by them, as well as the number of features and samples and the relevant attributes which should be selected by the feature selection methods. Notice whilst the main characteristic of each dataset is emphasized, it can have other characteristics too.

A.2.2.1 CorrAL

The CorrAL dataset [5] has six binary features (i.e., $f_1, f_2, f_3, f_4, f_5, f_6$), and its class value is $(f_1 \wedge f_2) \vee (f_3 \wedge f_4)$. Feature f_5 is irrelevant and f_6 is correlated to the class label by 75%.

CorrAL-100 [6] was constructed by adding 93 irrelevant binary features to the previous CorrAL dataset. The data for the added features were generated randomly. Both datasets (CorrAL and CorrAL-100) have 32 samples that are formed by considering all possible values of the four relevant features and the correlated one (2^5). The correct behavior for a given feature selection method is to select the four relevant features and to discard the irrelevant and correlated ones. The correlated feature is redundant if the four relevant features are selected, and, besides, it is correlated

Table A.1: Summary of the synthetic datasets used. "Corr." stands for "Correlation"

Dataset	No. of features	No. of samples	Relevant features	Corr.	Noise	Nonlinear	No. feat >> No. samples
Corral	6	32	1–4	✓			
Corral-100	99	32	1–4	✓			✓
Led-25	24	50	1–7		✓		
Led-100	99	50	1–7		✓		✓
Madelon	500	2400	1–5		✓	✓	
Monk1	6	122	1,2,5			✓	
Monk2	6	122	1–6			✓	
Monk3	6	122	2,4,5		✓		
Parity3+3	12	64	1–3			✓	
SD1*	4020	75	G_1, G_2				✓
SD2*	4040	75	$G_1 - -G_4$				✓
SD3*	4060	75	$G_1 - -G_6$				✓
XOR-100	99	50	1,2			✓	✓

* G_i means that the feature selection method must select only one feature within the *i*th group of features.

to the class label by 75%, so if one applies a classifier after the feature selection process, 25% error will be obtained.

A.2.2.2 XOR-100

XOR-100 [6] has two relevant binary features and 97 irrelevant binary features (randomly generated). The class attribute takes binary values and the dataset consists of 50 samples. Features f_1 and f_2 are correlated with the class value with XOR operation (i.e., class equals $f_1 \oplus f_2$). This is a hard dataset for the sake of feature selection because of the small ratio between number of samples and number of features and due to its nonlinearity (unlike the CorrAL dataset, which is a multi-variate dataset).

A.2.2.3 Parity3+3

The parity problem is a classic problem where the output is $f(x_1,\dots,x_n) = 1$ if the number of $x_i = 1$ is odd and $f(x_1,\dots,x_n) = 0$ otherwise. The Parity3+3 dataset is a modified version of the original parity dataset. The target concept is the parity of three bits. It contains 12 features among which three are relevant, another 3 are redundant (repeated) and another six are irrelevant (randomly generated).

A.2.2.4 The Led Problem

The Led problem [7] is a simple classification task that consists of, given the active leds on a seven segment display, identifying the digit that the display is representing. Thus, the classification task to be solved is described by seven binary attributes (see Figure A.1) and ten possible classes available ($C = \{0,1,2,3,4,5,6,7,8,9\}$). A 1 in an attribute indicates that the led is active, and a 0 indicates that it is not active.

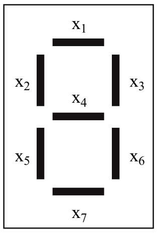


Fig. A.1: Led scheme

Two versions of the Led problem will be used: the first one, Led25, adding 17 irrelevant attributes (with random binary values), and the second one, Led100, adding 92 irrelevant attributes. Both versions contain 50 samples. The small number of samples was chosen because we are interested in dealing with datasets with a large number of features and a small sample size. Also, different levels of noise (altered inputs) have been added to the attributes of these two versions of the Led dataset: 2%, 6%, 10%, 15% and 20%. In this manner, the tolerance to different levels of noise of the feature selection methods tested will be checked. Note that, as the attributes take binary values, adding noise means assigning an incorrect value to the relevant features.

A.2.2.5 The Monk Problems

The MONK problems [8] rely on an artificial robot domain, in which robots are described by six different discrete attributes $(x_1,\ldots,x_6)$. The learning task is a binary classification task. The logical description of the class of the Monk problems is the following:

- Monk1: $(x_1 = x_2) \vee (x_5 = 1)$
- Monk2: $(x_n = 1)$ exactly two $n \in 1,2,3,4,5,6$
- Monk3: $(x_5 = 3 \wedge x_4 = 1) \vee (x_5 \neq 4 \wedge x_2 \neq 3)$

In the case of Monk3, among the 122 samples, 5% are misclassifications, i.e., noise in the target.

A.2.2.6 SD1, SD2 and SD3

These three synthetic datasets (SD1, SD2 and SD3) [9] are challenging problems because of their large number of features (around 4,000) and the small number of samples (75), besides a large number of irrelevant attributes.

SD1, SD2 and SD3 are three-class datasets with 75 samples (each class containing 25 samples) generated based on the approach described by [10]. Each synthetic dataset consists of both relevant and irrelevant features. The relevant features in each dataset are generated from a multivariate normal distribution using mean and covariance matrices [9]. Besides, 4,000 irrelevant features are added to each dataset, where 2,000 are drawn from a normal distribution of N(0,1) and the other 2,000 are sampled with a uniform distribution $U[-1,1]$. It is necessary to introduce some new definitions of multiclass relevancy features: full class relevant (FCR) and partial class relevant (PCR) features. Specifically, FCR denotes genes (features) that serve as candidate biomarkers for discriminating between all cancer types. However, PCR are genes (features) that distinguish subsets of cancer types.

SD1 is designed to contain only 20 FCR and 4,000 irrelevant features. Two groups of relevant genes are generated from a multivariate normal distribution, with 10 genes in each group. Genes in the same group are redundant with each other and the optimal gene subset for distinguishing the three classes consists of any two relevant genes from different groups.

SD2 is designed to contain 10 FCR, 30 PCR, and 4,000 irrelevant features. Four groups of relevant, i.e., FCR and PCR, genes are generated from a multivariate normal distribution, with 10 genes in each group. Genes in each group are redundant to each other, and in this dataset, only genes in the first group are FCR genes while genes in the three last groups are PCR genes. The optimal gene subset to distinguish all the three classes consists of four genes, one FCR gene from the first group and three PCR genes each from one of the three remaining groups.

SD3 has been designed to contain only 60 PCR and 4,000 irrelevant features. Six groups of relevant genes are generated from a multivariate normal distribution, with 10 genes in each group. Genes in the same group are designed to be redundant to each other and the optimal gene subset to distinguish all the three classes thus consists of six genes with one from each group.

It has to be noted that the easiest dataset in order to detect relevant features is SD1, since it contains only FCR features and the hardest one is SD3, due to the fact that it contains only PCR genes, which are more difficult to detect.

A.2.2.7 Madelon

The Madelon dataset [11] is a two-class problem originally proposed in the NIPS 2003 feature selection challenge. The relevant features are situated on the vertices of a five-dimensional hypercube. Five redundant features were added, obtained by multiplying the useful features by a random matrix. Some of the previously defined features were repeated to create 10 more features. The other 480 features are drawn from a Gaussian distribution and labeled randomly. This dataset presents high dimensionality both in number of features and in number of samples and the data were distorted by adding noise, flipping labels, shifting and rescaling. For all these reasons, it conforms to a hard dataset for the sake of feature selection.

A.2.3 DNA Microarray Datasets

DNA microarray data classification is a serious challenge for machine learning researchers due to its high dimensionality and small sample size. Typical values are around 10,000 gene expressions and 100 or less tissue samples. For this reason, these types of datasets are usually employed to test the efficiency of a feature selection method. Some datasets were originally divided to training and test sets whilst others only have a training set. On the one hand, Table A.2 shows the main characteristics of the datasets employed in this book having a unique training set (for example, for applying five-fold cross validation). On the other hand, Table A.3 presents the characteristics of the datasets used with separated training and test sets. All the datasets described in this section are available for download in [12] and [13].

Table A.2: Dataset description for binary microarray datasets

Dataset	Attributes	Samples	Distribution
Brain	12625	21	33–67%
CNS	7129	60	35–65%
Colon	2000	62	35–65%
DLBCL	4026	47	49–51%
GLI	22283	85	31–69%
Ovarian	15154	253	36–64%
SMK	19993	187	48–52%

Table A.3: Dataset description for binary microarray datasets with train and test sets

Dataset	Attributes	Samples		Train	Test
		Train	Test	distribution	distribution
Breast	24481	78	19	44–56%	37–63%
Prostate	12 600	102	34	49–51%	26–74%

A.3 Validation Techniques

To evaluate the goodness of the selected set of features, it is necessary to have an independent test set with data which have not been seen by the feature selection method. In some cases, the data come originally distributed into training and test sets, so the training set is usually employed to perform the feature selection process and the test set is used to evaluate the appropriateness of the selection. However, not all the datasets come originally partitioned. To overcome this issue, several validation techniques exist, and in the following we describe those used in this book.

A.3.1 *k-Fold Cross-validation*

This is one of the most famous validation techniques [14]. The data (D) is partitioned into k nonoverlapping subsets $D_1, \ldots, D_k$ of roughly equal size. The learner is trained on $k-1$ of these subsets combined together and then applied to the remaining subset to obtain an estimate of the prediction error. This process is repeated in turn for each of the k subsets, and the cross-validation error is given by the average of the k estimates of the prediction error thus obtained. In the case of feature selection, note that with this method there will be k subsets of selected features. A common practice is to merge the k different subsets (either by union or by intersection) or to keep the subset obtained in the fold with the best classification result.

A.3.2 *Leave-One-Out Cross-validation*

This is a variant of k-fold cross validation where k is the number of samples [14]. A single observation is left out each time.

A.3.3 Bootstrap

This is a general resampling strategy [15]. A *bootstrap sample* consists of n being the number of samples equally likely to be drawn, with replacement, from the original data. Therefore, some of the samples will appear multiple times, whereas others will not appear at all. The learner is designed on the bootstrap sample and tested on the left-out data points. The error is approximated by a sample mean based on independent replicates (usually between 25 and 200). Some famous variants of this method exist, such as *balanced bootstrap* or *0.632 bootstrap* [16]. As in the previous methods, there will be as many subsets of features as repetitions of the method.

A.3.4 Holdout Validation

This technique consists of randomly splitting the available data into a disjoint pair training test [14]. A common partition is to use 2/3 for training and 1/3 for testing. The learner is designed based on the training data and the estimated error rate is the proportion of errors observed in the test data. This approach is usually employed when some of the datasets in a study come originally divided into training and test sets whilst others do not. In contrast to other validation techniques, a unique set of selected features is obtained.

A.4 Statistical Tests

When performing several executions of a method, different results are obtained (e.g., after applying a k-fold cross validation). In this situation, statistical tests may be performed to check if there are significant differences among the medians for each method. In this book, the most used statistical methods were Kruskal-Wallis [17] and a multiple comparison procedure (Tukey's) [18]. The experimental procedure is detailed in the following:

1. A Kruskal-Wallis test is applied to check if there are significant differences among the medians for each model. In this book, we have opted for a level of significance $\alpha = 0.05$.
2. If the nonparametric Kruskal-Wallis test is significant, it means that at least a model exists that is better than the others. In this case, it is necessary to perform a multiple comparison procedure to figure out which model is the best.
3. Finally, the set of models with performance that is not significantly worse than the best is obtained. Among the models in this set, the simplest one should be selected.

A.5 Discretization Algorithms

Many filter algorithms are shown to work on discrete data [19]. In order to deal with numeric attributes, a common practice for those algorithms is to discretize the data before conducting feature selection. For both users and experts, discrete features are easier to understand, use, and explain and discretization can make learning more accurate and faster [20]. In general, the results obtained (decision trees, induction rules) by using discrete features are usually more compact, shorter and more accurate than those by using continuous ones; hence the results can be more closely examined, compared, used and reused. In addition to the many advantages of using discrete data over continuous data, a suite of classification learning algorithms can only deal with the former.

In essence, the process of discretization [21] involves the grouping of continuous values into a number of discrete intervals. However, the decisions of which continuous values to group together, how many intervals to generate, and thus where to position the interval cut points on the continuous scale of attribute values are not always identical for the different discretization methods. The discretizers used in this book are subsequently described. Previously, some notation considerations are given: given a numeric attribute X_i and n training instances for which the values of X_i are known, the minimum and the maximum values are v_{min} and v_{max} respectively. All the discretization methods first sort the values into ascending order.

A.5.0.1 Entropy Minimization Discretization, EMD

This popular method was created by Fayyad and Irani [22]. EMD evaluates as a candidate cut point the midpoint between each successive pair of the sorted values. For evaluating each candidate cut point, the data are discretized into two intervals and the resulting class information entropy is calculated. A binary discretization is determined by selecting the cut point for which the entropy is minimal amongst all candidates. The binary discretization is applied recursively, always selecting the best cut point. A minimum description length criterion (MDL) is applied to decide when to stop discretization.

A.5.0.2 Proportional *k*-Interval Discretization, PKID

PKID is a method created by Yang et al. [23]. The idea behind PKID is that discretization bias and variance relate to interval size and interval number. This strategy seeks an appropriate trade-off between the bias and variance of the probability estimation by adjusting the number and size of intervals to the number of training instances. The following compromise is adopted: given a numeric attribute, supposing we have n training instances with known values for the attribute, we discretize it into $\sqrt{n}$ intervals, with $\sqrt{n}$ instances in each interval. Thus, we give equal weight to both bias and variance management. Further, with n increasing, both the number and size of intervals increase correspondingly, which means discretization can decrease both the bias and variance of the probability estimation. This is very desirable, because if a numeric attribute has more instances available, there is more information about it. PKID has greater capacity to take advantage of the additional information inherent in large volumes of training data.

A.6 Classification Algorithms

If we are dealing with real datasets, the relevant features are not known a priori. Therefore, it is necessary to use a classification algorithm to evaluate the performance of the feature selection, focusing on the classification accuracy. Unfortunately, the class prediction depends also on the classification algorithm used, so when testing a feature selection method, a common practice is to use several classifiers to obtain results as classifier-independent as possible. This section describes the most common classification algorithms which are used throughout this book. Notice that some of them only can work with categorical features, whereas others require numerical attributes. In the first case, the problem is often solved by discretizing the numerical features. In the second case, it is common to use a conversion method which assigns numerical values to the categorical features.

A.6.1 Support Vector Machine, SVM

A Support Vector Machine [24] is a learning algorithm typically used for classification problems (text categorization, handwritten character recognition, image classification, etc.). More formally, a support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (so-called functional margin), since, in general, the larger the margin, the lower the generalization error of the classifier. In its basic implementation, it can only work with numerical data and binary classes.

A.6.2 Proximal Support Vector Machine, PSVM

This method classifies points, assigning them to the closest of two parallel planes (in input or feature space) that are pushed as far apart as possible [25]. The difference with a Support Vector Machine (SVM) is that PSVM classifies points by assigning them to one of two disjoint half-spaces. The PSVM leads to an extremely fast and simple algorithm by generating a linear or nonlinear classifier that merely requires the solution of a single system of linear equations.

A.6.3 C4.5

C4.5 is a classifier developed by [26], as an extension of the ID3 algorithm (Iterative Dichotomiser 3). Both algorithms are based on decision trees. A decision tree classifies a pattern building a descending filtering of itself until finding a leaf, which points to the corresponding classification. One of the improvements of C4.5 with respect to ID3 is that C4.5 can deal with both numerical and symbolic data. In order to handle continuous attributes, C4.5 creates a threshold and, depending on the value that takes the attribute, the set of instances is divided.

A.6.4 Naive Bayes, NB

A naive Bayes classifier [27] is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. This classifier assumes that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature, given the class variable. A naive Bayes classifier considers each of the features to contribute independently to the probability that a sample belongs to a given class, regardless of the presence or absence of the other features. Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In fact, naive Bayes classifiers are simple, efficient and robust to noise and irrelevant attributes. However, they can only deal with symbolic data, although discretization techniques can be used to preprocess the data.

A.6.5 k-Nearest Neighbors, k-NN

k-nearest neighbor [28] is a classification strategy that is an example of a "lazy learner." An object is classified by a majority vote from its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors (where k is some user-specified constant). If $k = 1$ (as is the case in the experiments in this book), then the object is simply assigned to the class of that single nearest neighbor. This method is more adequate for numerical data, although it can also deal with discrete values.

A.6.6 One-Layer Feedfoward Neural Network, One-Layer NN

This algorithm consists of training a single-layer feedfoward neural network using a new supervised learning method proposed by [29]. The method is based on the use of an alternative cost function that measures the errors before the nonlinear activation functions instead of after them, as is normally the case. As an important consequence, the solution can be obtained easily and rapidly because the new cost function is convex. Therefore, the absence of local minima is assured in this situation.

A.7 Evaluation Measures

In order to evaluate the behavior of the feature selection methods after applying a classifier, several evaluation measures need to be defined.

- *True positive (TP)*: percentage of positive examples correctly classified as so.
- *False positive (FP)*: percentage of negative examples incorrectly classified as positive.
- *True negative (TN)*: percentage of negative examples correctly classified as so.
- *False negative (FN)*: percentage of positive examples incorrectly classified as negative.
- $Sensitivity = \frac{TP}{TP+FN}$
- $Specificity = \frac{TN}{TN+FP}$
- $Accuracy = \frac{TN+TP}{TN+TP+FN+FP}$
- $Error = \frac{FN+FP}{TN+TP+FN+FP}$

A.7.1 Multiple-Criteria Decision-Making

Multiple-criteria decision-making (MCDM) [30] is focused on evaluating classifiers from different aspects and producing rankings of them. A multi-criteria problem is formulated using a set of alternatives and criteria. Among many MCDM methods that have been developed up to now, *technique for order of preference by similarity to ideal solution* (TOPSIS) [31] is a well-known method that will be used. TOPSIS finds the best algorithms by minimizing the distance to the ideal solution whilst maximising the distance to the anti-ideal one. The extension of TOPSIS proposed by [32] is used in this research.

References

1. MATLAB. *version 8.1.0.604 (R2013a)*. The MathWorks Inc., 2013.
2. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I.H. The Weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.
3. Bache, K. and Lichman, M. UCI Machine Learning Repository, 2013. University of California, Irvine, School of Information and Computer Sciences. http://archive.ics.uci.edu/ml .
4. Belanche, L. A. and González, F. F. Review and evaluation of feature selection algorithms in synthetic problems. *arXiv preprint arXiv:1101.2320*, 2011.
5. John, G. H., Kohavi, R. and Pfleger, K. Irrelevant features and the subset selection problem. In *International Conference on Machine Learning*, pages 121–129, 1994.
6. Kim, G., Kim, Y., Lim, H. and Kim, H. An MLP-based feature subset selection for HIV-1 protease cleavage site analysis. *Artificial Intelligence in Medicine*, 48(2):83–89, 2010.
7. Breiman, L. *Classification and Regression Trees*. CRC Press, 1993.
8. Thrun, S. B., Bala, J. W., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., De Jong, K. A., Dzeroski, S., Fisher, D. H., Fahlman, S. E. and others. The monk's problems a performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Carnegie Mellon University, 1991.
9. Zhu, Z., Ong, Y. S. and Zurada, J. M. Identification of full and partial class relevant genes. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(2):263–277, 2010.
10. Díaz-Uriarte, R. and Alvarez De Andres, S. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7(1):3, 2006.
11. Guyon, I., Gunn, S., Nikravesh, M. and Zadeh, L. *Feature Extraction: Foundations and Applications*. Springer, 2006.
12. Broad Institute. Cancer Program Data Sets. http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi
13. Kent Ridge Bio-Medical Dataset. http://datam.i2r.a-star.edu.sg/datasets/krbd
14. Bramer, M. *Principles of Data Mining*. Springer, 2007.
15. Efron, B. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, pages 1–26, 1979.
16. Efron, B. and Tibshirani, R. *An Introduction to the Bootstrap*, volume 57. Chapman & Hall/CRC, 1993.
17. Wolfe, D. A. and Hollander, M. *Nonparametric Statistical Methods*, 1973.
18. Hsu, J. C. *Multiple Comparisons: Theory and Methods*. CRC Press, 1996.
19. Liu, H. and Setiono, R. Feature Selection via Discretization. *Journal of IEEE Transactions on Knowledge and Data Engineering*, 9(4):642-645, 1997.
20. Liu, H., Hussain, F., Tan, C. L. and Dash, M. Discretization: An Enabling Technique. *Data Mining and Knowledge Discovery Journal*, 6(4):393-423, 2002.
21. Janssens, D., Brijs, T., Vanhoof, K. and Wets, G. Evaluating the performance of cost-based discretization versus entropy and error-based discretization. *Computers and Operations Research Journal*, 33(11):3107-3123, 2006.
22. Fayyad, U. M. and Irani, K. B. Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *13th International Joint Conference on Artificial Intelligence*, pages 1022-1029, 1993.
23. Yang, Y. and Webb, G. I. Proportional *k*-Interval Discretization for Naive-Bayes Classifiers. In *12th European Conference on Machine Learning*, pages 564–575, 2001.
24. Vapnik, V. N. *Statistical Learning Theory*. Wiley, 1998.
25. Fung, G. and Mangasarian, O. L. Proximal support vector machine classifiers. In *7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 77–86. ACM, 2001.
26. Quinlan, J. R. *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

27. Rish, I. An empirical study of the naive Bayes classifier. In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, volume 3, pages 41–46, 2001.
28. Aha, D. W., Kibler, D. and Albert, M. K. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.
29. Castillo, E., Fontenla-Romero, O., Guijarro-Berdiñas, B. and Alonso-Betanzos, A. A global optimum approach for one-layer neural networks. *Neural Computation*, 14(6):1429–1449, 2002.
30. Zeleny, M. and Cochrane, J. L. *Multiple Criteria Decision Making*. McGraw-Hill, 1982.
31. Hwang, C. L. and Yoon, K. *Multiple Attribute Decision Making*. Springer, 1981.
32. Olson, D. L. Comparison of weights in TOPSIS models. *Mathematical and Computer Modelling*, 40(7):721–727, 2004.