Doctoral Thesis

Simple but not simplistic: reducing the complexity of machine learning methods

Laura Morán Fernández

2020

UNIVERSIDADE DA CORUÑA

Simple but not simplistic: reducing the complexity of machine learning methods

Laura Morán Fernández

Doctoral Thesis

March 2020

PhD Advisors:
Amparo Alonso Betanzos Verónica Bolón Canedo

PhD Program in Computer Science

UNIVERSIDADE DA CORUÑA

Amparo Alonso Betanzos
Catedrática de Universidad
Departamento de Ciencias de la Computación y Tecnologías de la Información
Universidade da Coruña

Verónica Bolón Canedo
Profesora ayudante doctor
Departamento de Ciencias de la Computación y Tecnologías de la Información
Universidade da Coruña

CERTIFICAN

Que la memoria titulada "*Simple but not simplistic: reducing the complexity of machine learning methods*" ha sido realizada por D. Laura Morán Fernández bajo nuestra dirección en el Departamento de Ciencias de la Computación y Tecnologías de la Información de la Universidade da Coruña, y concluye la Tesis Doctoral que presenta para optar al grado de Doctor en Ingeniería Informática con la Mención de Doctor Internacional.

En A Coruña, a 5 de Marzo de 2020

Fdo.: Amparo Alonso Betanzos
Directora de la Tesis Doctoral

Fdo.: Verónica Bolón Canedo
Directora de la Tesis Doctoral

Fdo.: Laura Morán Fernández
Autora de la Tesis Doctoral

Para Alba

Acknowledgments

Keep Ithaka (PhD) always in your mind. Arriving there is what you're destined for. But don't hurry the journey at all. Better if it lasts for years (five), so you're old by the time you reach the island (deposit), wealthy with all you've gained on the way, not expecting Ithaka (PhD) to make you rich.

This long, tricky and full of adventures journey would not have been possible without the help, support and guidance of many people. Firstly, I would like to express my sincere gratitude towards my advisors Verónica and Amparo, who have taught me how to do research, to write and to communicate my ideas. I would like to thank them for believing in my work and giving me this amazing opportunity.

I would like to say thanks to all my past and present colleagues in the LIDIA group, specially Sasi, for their fellowship, support, and all the coffee breaks and dinners we shared together. A special thanks goes to my mates from another lab: Nuria, Jorge and Manu, who were always there for me in the bad times and in the good ones.

I would like to thank Kostas Sechidis and Gavin Brown for their insights on my work, and all the friends I made in the MLO group. Thanks for bringing me the "sun" to Liverpool (or a city nearby).

Nobody has been more important to me in the pursuit of this thesis than the members of my family. I want to say thanks to my mum and dad for all the love and strength they gave me in live and for always encouraging me to pursue my dreams; to my little psychologist Rafa for his unconditional support; and to my heroes Diana, Ana and Paco.

Last but not least, I would like to thank all my friends, in León, Coruña, Bergen and elsewhere, for providing me with some much needed distraction and reminding me of the world beyond Machine Learning! And thanks to Pablo, his patience and support during these last months cannot be underestimated.

Laura Morán Fernández

Science is not a boy's game,
it's not a girl's game.
It's everyone's game.
It's about where we are and where we're going.

Nichelle Nichols

Resumo

A chegada do *Big Data* e a explosión do Internet das cousas supuxeron un gran reto para os investigadores en Aprendizaxe Automática, facendo que o proceso de aprendizaxe sexa mesmo máis complexo. No mundo real, os problemas da aprendizaxe automática xeralmente teñen complexidades inherentes, como poden ser as características intrínsecas dos datos, o gran número de mostras, a alta dimensión dos datos de entrada, os cambios na distribución entre o conxunto de adestramento e test, etc. Todos estes aspectos son importantes, e requiren novos modelos que poidan facer fronte a estas situacións. Nesta tese, abordáronse todos estes problemas, tratando de simplificar o proceso de aprendizaxe automática no escenario actual. En primeiro lugar, realízase unha análise de complexidade para observar como inflúe esta na tarefa de clasificación, e se é posible que a aplicación dun proceso previo de selección de características reduza esta complexidade. Logo, abórdase o proceso de simplificación da fase de aprendizaxe automática mediante a filosofía *divide e vencerás*, usando un enfoque distribuído. Seguidamente, aplicamos esa mesma filosofía sobre o proceso de selección de características. Finalmente, optamos por un enfoque diferente seguindo a filosofía do *Edge Computing*, a cal permite que os datos producidos polos dispositivos do Internet das cousas se procesen máis preto de onde se crearon. Os enfoques propostos demostraron a súa capacidade para reducir a complexidade dos métodos de aprendizaxe automática tradicionais e, polo tanto, espérase que a contribución desta tese abra as portas ao desenvolvemento de novos métodos de aprendizaxe máquina máis simples, máis robustos, e máis eficientes computacionalmente.

Resumen

La llegada del *Big Data* y la explosión del Internet de las cosas han supuesto un gran reto para los investigadores en Aprendizaje Automático, haciendo que el proceso de aprendizaje sea incluso más complejo. En el mundo real, los problemas de aprendizaje automático generalmente tienen complejidades inherentes, como pueden ser las características intrínsecas de los datos, el gran número de muestras, la alta dimensión de los datos de entrada, los cambios en la distribución entre el conjunto de entrenamiento y test, etc. Todos estos aspectos son importantes, y requieren nuevos modelos que puedan hacer frente a estas situaciones. En esta tesis, se han abordado todos estos problemas, tratando de simplificar el proceso de aprendizaje automático en el escenario actual. En primer lugar, se realiza un análisis de complejidad para observar cómo influye ésta en la tarea de clasificación, y si es posible que la aplicación de un proceso previo de selección de características reduzca esta complejidad. Luego, se aborda el proceso de simplificación de la fase de aprendizaje automático mediante la filosofía *divide y vencerás*, usando un enfoque distribuido. A continuación, aplicamos esa misma filosofía sobre el proceso de selección de características. Finalmente, optamos por un enfoque diferente siguiendo la filosofía del *Edge Computing*, la cual permite que los datos producidos por los dispositivos del Internet de las cosas se procesen más cerca de donde se crearon. Los enfoques propuestos han demostrado su capacidad para reducir la complejidad de los métodos de aprendizaje automático tradicionales y, por lo tanto, se espera que la contribución de esta tesis abra las puertas al desarrollo de nuevos métodos de aprendizaje máquina más simples, más robustos, y más eficientes computacionalmente.

Abstract

The advent of Big Data and the explosion of the Internet of Things, has brought unprecedented challenges to Machine Learning researchers, making the learning task more complex. Real-world machine learning problems usually have inherent complexities, such as the intrinsic characteristics of the data, large number of instances, high input dimensionality, dataset shift, etc. All these aspects matter, and call for new models that can confront these situations. Thus, in this thesis, we have addressed all these issues, simplifying the machine learning process in the current scenario. First, we carry out a complexity analysis to see how it influences the classification models, and if it is possible that feature selection might result in a decrease of that complexity. Then, we address the process of simplifying learning with the *divide-and-conquer* philosophy of the distributed approach. Later, we aim to reduce the complexity of the feature selection preprocessing through the same philosophy. Finally, we opt for a different approach following the current philosophy Edge computing, which allows the data produced by Internet of Things devices to be processed closer to where they were created. The proposed approaches have demonstrated their capability to reduce the complexity of traditional machine learning algorithms, and thus it is expected that the contribution of this thesis will open the doors to the development of new machine learning methods that are simpler, more robust, and more computationally efficient.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The "big data" phenomenon is unfolding before our eyes and its life-changing nature is unquestionable. Over the past 20 years, data has increased in volume, variety and velocity in various disciplines. Alongside advances in storage technology, scientific research laboratories and enterprises have become interested in the high potential of all these data for deriving relevant information. These data contain buried within it knowledge that can be critical to a enterprise's growth or decline, knowledge that could lead to important discoveries in science, knowledge that could enable us accurately to predict natural disasters, knowledge that could enable us to identify the causes of and possible treatments for lethal illnesses. Thus, Machine Learning (ML) methods have become indispensable in order to extract useful information from huge amounts of otherwise meaningless data. In a society that needs to deal with big data, there is an urgent need for new analysis and processing tools.

1.1. Exploring the complexity in machine learning

Apart from the obvious increase in data size —both in number of samples and features—, machine learning techniques have to be capable of dealing also with other challenges given by the huge variety of data, as well as its changing nature and the Internet of Things explosion. Therefore, in the era of big data, the complexity of a machine learning problem takes multiple forms. Some of such forms are:

- **INTRINSIC CHARACTERISTICS OF THE DATA.** During the last few decades, tremendous progress has been made on developing and refining machine learning algorithms. However, automatic learning still appears to be far from reach in daily tasks. Therefore, when the classifiers are not perfect, is it a deficiency of the algorithms by design?, or is it a difficulty intrinsic to the classification task?

  The Bayes error is a well-known description of the intrinsic complexity of a classification problem, which can be computed if complete knowledge is given on the probability distribution of each class in the feature space [73, 114]. The universal distribution assigns high probabilities to simple patterns, and thus implicitly prefers simple hypothesis. But empirical studies suggest that, from the perspectives of classification algorithms given limited training data, problems can be complex for different reasons even if the same Bayes error is achieved. This results in strongly data-dependent performances of classifiers. Intrinsic data characteristics affecting accuracy can be, for example, the shapes of the classes: the shape of the decision boundary, the amount of overlap among the classes, the proximity of two classes, and the number of informative samples available for training.

  A prerequisite for setting appropriate expectations on classification performance is to understand the complexity of a specific problem arising from an application. The work from Ho and Basu [10] was seminal in analyzing the complexity of a classification problem by using descriptors extracted from a learning dataset. Given that there is none ML technique that can consistently obtain the best performance for every classification problem [126], this type of analysis allows to find out whether, or to which extent, patterns exist in the data. It is also useful to obtain guidance selecting specific classification techniques. We consider that this is one of the keys for further advances in classification.

- **LARGE NUMBER OF INSTANCES.** Databases continue to increase in size. Between the dawn of time up to 2003 humanity generated a total of 5 exabytes of data and by 2008 this figure had tripled to 14.7 exabytes

[19]. Nowadays 5 exabytes of data is produced every 2 days, and the pace of production continues to rise.

Under the explosive increase of global data, the term of big data is mainly used to describe datasets that could not be perceived, acquired, managed, and processed by traditional information technology and software/hardware tools within a tolerable time [28]. Thus, this new big data scenario offers both opportunities for discovering new values, helps us to gain an in-depth understanding of the hidden values, and also incurs new challenges, e.g. how to effectively organize and manage such datasets.

Most existing learning algorithms were developed when datasets sizes were much smaller, but nowadays different solutions are required for large-scale learning problems. Small-scale learning problems are subject to the usual approximation-estimation trade-off, but this trade-off is more complex in the case of large-scale learning problems, not only because of accuracy but also due to the computational cost of the learning algorithm. Moreover, since most algorithms rely on in-memory data structures, these algorithms are useless when the entire dataset does not fit in system memory. This means that temporary results will have to be written out to disk or the dataset will have to be divided into partitions small enough to be processed in memory, entailing further scans.

The preferred way to effectively process such datasets is to combine the distributed storage and bandwidth of a cluster of machines. Several paradigms for performing distributed learning have emerged in the last decade, such as MapReduce [36], Hadoop [5], Spark [6] and Flink [4]. Such frameworks combine the ability to use high-capacity storage and execution platforms with programming via simple, naturally parallelizable language primitives.

- **HIGH INPUT DIMENSIONALITY.** In some applications, data samples are represented by a very large number of features. Machine learning scenarios such as DNA microarray analysis, image classification, face recognition or text classification can easily have several millions of input features, far exceeding the range of 10-1000 considered common until recently [12]. Thus, the advent of big data has raised unprecedented challenges for researchers. The high input dimensionality not only incurs into unbearable memory requirements and high computational cost in training, but also deteriorates the generalization ability of ML algorithms because of the *curse of dimensionality* [13]. This issue arises when analyzing and organizing data in high-dimensional spaces, and that does not usually occur in low-dimensional settings. To avoid this *curse of dimensionality*, feature selection—defined as the process of identifying the relevant features from the training data—is advisable.

Traditionally, feature selection methods have been designed to run in a centralized computing environment. In order to cope with such an unprecedented number of features posed by the explosion of big data, there is a growing need for scalable yet efficient feature selection methods, given than existing methods are likely to prove inadequate.

- **DATASET SHIFT.** Systems based on machine learning techniques often face a major challenge when applied "in the wild": The conditions under which the system was developed might differ from those in which we will use it, caused by intrinsic sample selection bias or inevitable non-stationary scenarios [42]. However, most supervised learning algorithms assume that training and test data follow the same probability distribution. In classification scenarios changes in class balance are often observed, e.g. the female-male ratio is almost fifty-fifty in the real-world (test set), whereas training samples collected in a research laboratory tend to be dominated by male data.

  This problem complicates the task of learning a model from data and requires special approaches, different from commonly used techniques, which treat arriving instances as equally important contributors to the final concept [121].

- **INTERNET OF THINGS EXPLOSION.** Due to the growth of wireless communication technology and to the cost reduction of electronic components, the number of Internet of Things (IoT) devices has exploded in recent years. These devices continuously generate zettabytes of data, which must be fed to a ML system to analyze information and make decisions. However, limitations in the computational capabilities of portable embedded systems—small memories and limited computing power—inhibit the implementation of most of the current ML algorithms on them.

To meet this demand, a new computing paradigm, Edge Computing [110], has emerged. Until recently, these millions of devices that conform the IoT just recorded data, and send them to the cloud or a computer center where it was processed to obtain information and knowledge from that data. Edge computing aims at changing this passive situation improving efficiency by allowing the nodes of the network or the very own devices to analyze the generated data. In this way, beside avoiding unnecessary network traffic the paradigm allows obtaining real-time process knowledge. There are also factors that will make this type of paradigm even easier in the future: the increasingly reduced cost of devices and sensors joins the increasing power of even modest devices. There are also industrial needs that contribute to betting on Edge Computing: in certain environments the only way to further optimize processes is to try to avoid communication with the cloud as much as possible. This allows to reduce latencies, consume less bandwidths—it is not necessary to send all the data to the cloud at all times—and immediately access analysis and evaluation of the state of all those sensors and devices.

There is another interesting advantage: security. The less data there is in a cloud environment, the less vulnerable is that environment if it is compromised. Of course, the security in those "micro data centers" should be taken care of properly. However, this does not mean that Cloud Computing environments disappear: both trends must contribute, and for example Edge Computing is more appropriate when above all speed and low latency are needed in those data that require remarkable computing power. For that reason, both scenarios are taken into account in this work.

Real-world machine learning problems usually have inherent complexities, such as intrinsic characteristics of the data, large number of instances, high input dimensionality, dataset shift, etc. All these aspects matter, and call for new models that can confront these situations. Thus, in this thesis, we have addressed all these issues, simplifying the machine learning process, which is currently even more complex due to the big data and IoT scenarios. To begin with, we carry out a complexity analysis to see how it influences the classification, and if it is possible that feature selection results in a decrease of that complexity. Then, we address the process of simplifying learning with the philosophy of the distributed approach. Later, we try to reduce the complexity of the feature selection through the same philosophy. Finally, we opt for a different approach following the current philosophy of computing close to the source (edge computing). To the best of our knowledge and, unlike learning approaches, feature selection techniques under computational constraints have not been implemented yet.

1.2. Structure of this thesis

This thesis can be divided into four different parts, that aim to find a solution to all the problems raised above:

1. **Data complexity.** This part of the thesis is committed with data complexity analysis, which enables an understanding of whether classification performance could be affected, not by algorithm limitations, but by intrinsic data characteristics. For showing how to confront this problem, we have chosen as an example microarray datasets. These datasets are characterized by high numbers of gene expressions combined with small sample sizes, and represent a particular challenge for machine learning researchers. This type of data also has other particularities that may negatively affect the generalization capacity of classifiers, such as overlaps between classes and class imbalance. Making use of several data complexity measures, we explored the connection between the intrinsic complexity of several microarray datasets and the empirical results obtained by four widely used classifiers, analyzing as well if feature selection reduces that complexity. Experimental results for 21 binary and multiclass datasets demonstrate that a correlation exists between microarray data complexity and the classification error rates.

2. **Distributed learning.** Traditional machine learning algorithms and, more specifically, data mining algorithms, do not scale well—memory demands and impracticable runtimes—, damaging performance and efficiency in the current big data scenario. Thus, we consider a distributed framework where training and test samples drawn from the same distribution are available, with the training instances spread across disjoint nodes. In this setting, a novel learning algorithm based on combining with different weights the outputs of classifiers

trained at each node is proposed. The weights depend on the distributional distance between each node and the test set in the feature space. Two different weighting approaches are introduced, which are referred to as per-Node Weighting (pNW) and per-Instance Weighting (pIW). While pNW assigns the same weight to all test instances at each node, pIW allows distinct weights for test instances differently represented at the node. By construction, our approach is particularly useful to deal with unbalanced nodes. Our methods require no communication between nodes, allowing for data privacy, independence of the kind of trained classifier at each node and maximum training speedup. In fact, our methods do not require retraining of the node's classifiers if available. Although a range of different combination rules are considered to ensemble the single classifiers, theoretical support for the optimality of using the sum rule is provided. Our experiments illustrate all of these properties and show that pIW produces the highest classification accuracies compared with pNW and the standard unweighted approaches.

3. **Distributed feature selection.** In the big data context, many datasets have a common characteristic, the large number of features. As a result, selecting the relevant features and ignoring the irrelevant and redundant features has become indispensable. However, similar to learning methods, when dealing with large amounts of data, most existing feature selection algorithms do not scale well, and their efficiency may significantly deteriorate to the point of becoming inapplicable. Moreover, data is often distributed in multiple locations, and it is not economic or legal to gather it in a single site. For these reasons, we propose a complexity reduction of the feature selection process following the distributed philosophy. Unlike existing procedures to combine the partial outputs obtained from each partition of data, we propose a merging process using the theoretical complexity of these feature subsets yielding better or comparable accuracy results, and a considerable reduction in computation time. The novel procedure tested in 11 datasets has proved to be useful, showing competitive results both in terms of runtime and classification accuracy.

4. **Feature selection under computational constraints.** Since wearable computing systems have grown in importance in the last years, there is an

increased interest in implementing machine learning algorithms with reduced precision parameters/computations. Not only learning, also feature selection, most of the times a mandatory preprocessing step in machine learning, is often constrained by the available computational resources. This part considers mutual information—one of the most common measures of dependence used in feature selection algorithms—with a limited number of bits. In order to test the procedure designed, we have implemented it in several well-known feature selection algorithms. Experimental results over several synthetic and real datasets demonstrate that low bit representations are sufficient to achieve performances close to that of double precision parameters and thus open the door for the use of feature selection in embedded platforms that minimize the energy consumption and carbon emissions.

Finally, the main conclusions and contributions of this thesis are summarized. Notice that Appendix A presents the methods used throughout this thesis and Appendix B reports additional results from our experiments.

Chapter 2

Data complexity

Throughout the 1990s, research into supervised classification has resulted in a rich set of classifier technologies for improving accuracy. One of the main problems of supervised classification is the generalization capacity of the learned classifier. Classification rules, typically derived from randomly selected training samples for each class, are used to test new instances and evaluate classification performance. However, since the accuracy of each classifier is strongly dependent on the characteristics of the data, an analysis of intrinsic data complexity would appear to be essential in order to select classification algorithms suitable for particular problems. Data complexity analysis, a relatively recent proposal by Ho and Basu [10], identifies data particularities which imply some difficulty for the classification task—such as overlaps between classes, class separability or decision boundary linearity—and also identifies relationships (correlations) with classifier accuracy. In our research we applied data complexity measures to microarray data given the intrinsic characteristics of these datasets.

Microarray technology is used to collect information from tissue and cell samples regarding gene expression differences that could be useful for diagnosing diseases. The classification of this type of data has been viewed as a particular challenge for machine learning (ML) researchers over the last 20 years, mainly due to the mismatch between dimensionality and sample size. The existence of many fields relative to few samples means that false positives findings due to chance are very likely in terms of both identifying relevant genes and building predictive models [99]. Moreover, several studies have demonstrated that most of the genes measured in a DNA microarray experiment do not actually contribute to efficient sample classification. To avoid this "curse of dimensionality", feature selection—defined as the process of identifying and removing irrelevant features from the training data—is advisable so as to identify the specific genes that enhance classification accuracy. As well as their extremely high dimensionality, microarray datasets have other properties that complicate the classification task [20, 106]. This type of theoretically complex dataset is thus quite interesting for a study aimed at addressing the issue of classifier choice.

We believe that the impact on problems of algorithm selection and parameter optimization could be significant for high dimensionality problems in general and for challenging applications like microarray data analysis in particular. Our aim, therefore, is to explore the connection between the intrinsic characteristics of microarray data and the classification performance of four widely used algorithms. To do this we observed changes in complexity measures obtained for both binary and multiclass datasets with and without feature selection.

2.1. Background

It is widely acknowledged that the prediction capacities of classifiers greatly depend on the available data. Several studies have explored the use of data complexity analysis to characterize data and to relate data characteristics to classifier performance. The most commonly employed data complexity measures are those proposed (as mentioned in the Introduction of this chapter) by Ho and Basu [10]. Bernadó-Mansilla and Ho [14], moreover, investigated the domain of competence of an accuracy-based classifier system (XCS), characterizing classification problem complexity using a set of geometrical descriptors. Sánchez et al. [107], using the Ho and Basu measures as reference, analyzed how training data complexity affected nearest neighbor (NN) classifiers. Mollineda et al. [85] extended some of the Ho and Basu measures to multiclass problems, analyzing two classic prototype selection algorithms and proposing Fisher's discriminant ratio as the most effective technique. Other authors [81], for demonstrating the potential bias of advocating the superiority of any given learner for a limited set of problems, proposed characterizing datasets using complexity measures, as they are helpful both in guiding experimental design and in explaining learner behavior. Luengo et al. [78] presented an automatic extraction method to determinate the domains of competence of a classifier using a set of data complexity measures.

Studies focusing on the complexity of microarray data are less common, but include Lorena et al. [77], who, using the Ho and Basu measures, investigated the particular characteristics of several microarray datasets that most impacted on the prediction ability of Support Vector Machine (SVM) classifiers; the same authors also used suitable feature selection for the data, demonstrating that this procedure could reduce the influence of data characteristics on classifier error rates. Okun et al. [92], in a novel approach based on using an ensemble of k-NN classifiers, also found positive dependence between complexity and error.

Several studies have shown that most genes measured in gene expression microarray data are not crucial to classification accuracy [48], as selecting a small number of discriminative genes ensures effective categorization of diseases [40, 125, 127]. For this reason, feature selection is receiving growing attention in gene selection for sample classification and is being increasingly used as preprocessing step in tackling microarray data: it not only removes redundant and irrelevant features but also helps biologists identify the underlying mechanisms relating gene expression to disease [20].

Our aim was to analyze and compare the link between microarray data complexity and classification performance with and without feature selection, thereby contributing with further insights in an area which has only been partially studied to date [77, 92]. We also analyze changes in complexity measures before and after applying feature selection so as to provide some insights into why classification accuracy is improved in some cases.

2.2. Data complexity measures

Data complexity analysis is a recent proposal aimed at representing data particularities that add complexity to classification tasks—such as overlaps between classes, separability and decision boundary linearity—and at identifying relationships (correlations) with classification performance. To analyze the theoretical complexity of the microarray datasets chosen for this research, we used the measures proposed by Ho and Basu [93, 10]. These measures are grouped according to the aspect of the data they focus on.

2.2.1. Measures of overlap in feature values from different classes

These measures, which focus on the capacity of features to separate instances from different classes, examine the range and spread of values in the dataset within each class and check for overlaps between different classes.

- Maximum Fisher's discriminant ratio (F1). It measures the overlap between the values of the features in different classes and is given by:

  $$rf = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \tag{2.1}$$

  where μ_1, μ_2 are the means and σ_1^2 and σ_2^2 are the variances of the two classes, in that feature dimension. We computed rf for each feature and take the maximum as the F1 measure. That is, F1 takes the values of the largest discriminant ratio among all the available features. This is consistent with the definition that if at least one feature discriminates the classes, the dataset can be considered simpler than if no such feature exists.

  Roughly, the F1 measure computes the ratio of inter-class to intra-class scatter for each feature. High values of the F1 measure indicates the existence of a feature for whose values there is a little overlap among the different classes. That is, it indicates the existence of a feature for which a hyperplane perpendicular to its axis can separate the classes fairly. Nonetheless, if the required hyperplane is oblique to the feature axes, F1 may not be able to reflect the underlying simplicity.

  Taking, for instance, the problem shown in Figure 2.1, the most discriminative feature would be f_1. F1 correctly indicates that the classes can be easily separable using this feature. Feature f_2, on the other hand, is non-discriminative, since its values for the two classes overlap, with the same mean and variance.

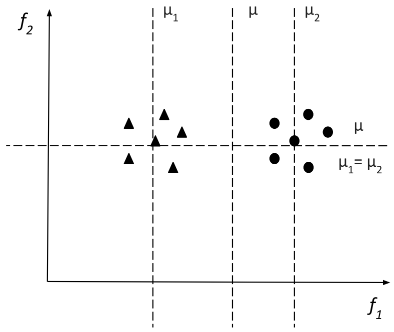


Figure 2.1: Example of F1 computation for a two-class problem.

Most of the data complexity measures were designed for two-class problems. A relatively simple way of adapting them to multiclass problems is to convert the original problem to several instances of two-class problems. Alternatively, the F1 measure can be extended to multiple classes according to their semantics ($mF1$):

$$rf = \frac{\sum_{i=1,j=1,i\neq j}^{c} p_i p_j (\mu_i - \mu_j)^2}{\sum_{i=1}^{c} p_i \sigma_i^2} \tag{2.2}$$

where μ_i, σ_i^2 and p_i are the mean, variance and proportion of the ith class, respectively. In practice the inverse of the Fisher ratio ($1/mF1$) is preferred, with a small complexity value representing a simple problem.

- Volume of overlap region (F2). This measure of the amount of overlap between bounding boxes of two classes is 0 if there is at least one feature in which the values of the two classes do not overlap. F2 can be determined by finding, for each feature f_i, its minimum and maximum values in the classes. The range of the overlapping interval is then calculated, normalized by the range of the values in both classes. Finally, the obtained values are multiplied:

$$F2 = \prod_{i}^{d} \frac{max(0, minmax(f_i) - maxmin(f_i))}{maxmax(f_i) - minmin(f_i)} \tag{2.3}$$

where

$$\begin{aligned}
minmax(f_i) &= MIN(max(f_i, c_1), max(f_i, c_2)),\\
maxmin(f_i) &= MAX(min(f_i, c_1), min(f_i, c_2)),\\
maxmax(f_i) &= MAX(max(f_i, c_1), max(f_i, c_2)),\\
minmin(f_i) &= MIN(min(f_i, c_1), min(f_i, c_2)).
\end{aligned}$$

and f_i is the $i = 1, ..., d$ feature for a d-dimensional problem and c_1 and c_2 refer to the two classes.

The numerator becomes zero when the per-class value ranges are disjoint for at least one feature. This implementation uses a correction that was made in Souto et al. [35] to the original definition of F2, which may yield negative values for non-overlapping feature ranges. The higher the F2 value, the greater amount of overlap between the problem classes. Thus, the problem's complexity is also higher. Figure 2.2 illustrates the region that F2 tries to capture (as the shared area), for a problem with two features and two classes.

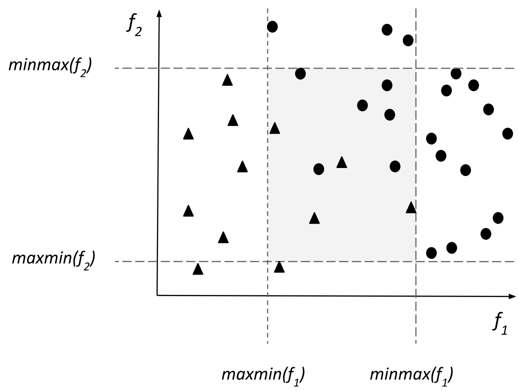


Figure 2.2: Example of overlapping region.

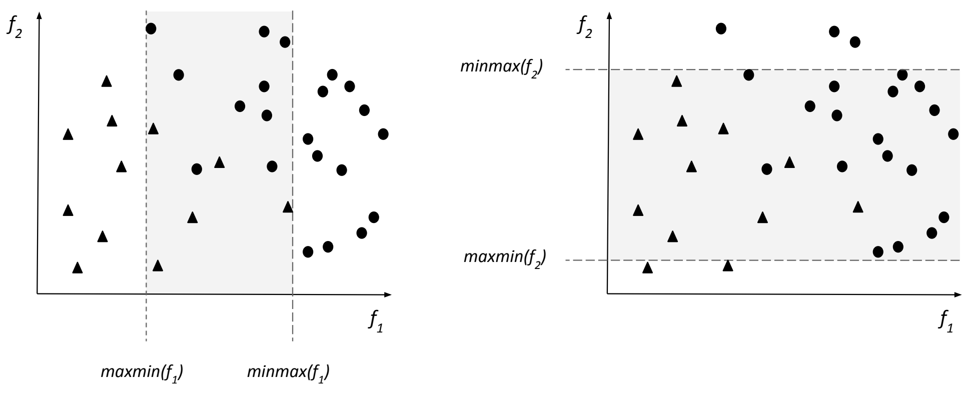
- Maximum (individual) feature efficiency (F3). In a procedure that progressively removes unambiguous points falling outside the overlapping region in each chosen dimension, the efficiency of each feature is defined as the fraction of all remaining points separable by that feature. To represent the contribution of the most useful feature, we use *maximum feature efficiency* as measure F3.

$$F3 = \max_{i=1}^{d} \frac{n - n_o(f_i)}{n}, \tag{2.4}$$

where $n_0(f_i)$ gives the number of samples that are in the overlapping region for feature f_i:

$$n_o(f_i) = \sum_{j=1}^{n} I(x_{ji} > maxmin(f_i) \wedge x_{ji} < minmax(f_i)), \tag{2.5}$$

and I is the indicator function, which returns 1 if its argument is true and 0 otherwise, while $maxmin(f_i)$ and $minmax(f_i)$ are the same as defined for F2. Figure 2.3 presents the computation of F3 for the same problem from Figure 2.2. While for feature f_1 the proportion of samples that are not in the overlapping region is $\frac{19}{33}$ (Figure 2.3(a)), for f_2 this proportion is $\frac{5}{33}$ (Figure 2.3(b)), resulting in a F3 value of $\frac{19}{33}$.



(a) (b)

Figure 2.3: Calculating F3 for the problem from Figure 2.2.

This measure considers only separating hyperplanes perpendicular to the features axes so, even for a linear problem, it may be less than 1 if the optimal separating hyper-plane is oblique.

2.2.1.1. Measures of class separability

These measures reflect to which extent classes are separable by estimating the length and linearity of class boundaries. They are motivated by the assumption that a linearly separable problem can be considered simpler than a problem requiring non-linear decision boundary. To obtain the linear classifier, Ho and Basu [10] suggested to solve an optimization problem proposed by Smith [112], while in Orriols-Puig et al. [94] a linear Support Vector Machine (SVM) is used instead. Here we adopt the SVM solution.

The hyperplane sought in the SVM formulation is the one which separates the samples from different classes with a maximum margin while minimizing training errors. This hyperplane is obtained by solving the following optimization problem:

$$\text{Minimize}_{w,b,E} = \frac{1}{2} \parallel w \parallel^2 + C\left(\sum_{i=1}^{n} \varepsilon_i\right)$$

$$\text{Subject to :} \begin{cases} y_i(w \cdot x_i + b) \geq 1 - \varepsilon_i, \\ \varepsilon_i \geq 0, i = 1, ...n \end{cases}$$

where C is the trade-off between the margin maximization, achieved by minimizing the norm of w, and the minimization of the training errors, modeled by ε. The hyperplane is given by $w \cdot x + b = 0$, where w is a weight vector a b is an offset value.

- Minimized sum of the error distance by linear programming (L1). This measure evaluates to which extent the training data is linearly separable. It returns the sum of the differences between a linear classifier predicted value and the actual class value. A zero value for L1 indicates that the problem is linearly separable and can be considered simpler than a problem for which a non-linear boundary is required.

  Given the SVM hyperplane, the error distance of the erroneous instances can be computed by summing up the ε_i values. For examples correctly classified, ε_i will be zero, which indicates the distance of the sample to the linear boundary otherwise.

$$L1 = \frac{1}{n} \sum_{i=1}^{n} \varepsilon_i \tag{2.6}$$

where the ε_i values are determined in the SVM optimization process.

Figure 2.4 presents an example of L1 application. After a linear boundary is obtained, the ε_i values of the misclassified examples (gray triangles) are summed up.

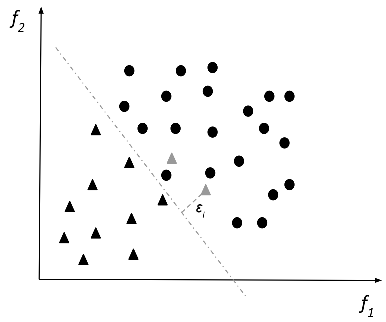


Figure 2.4: Example of L1 and L2 computation.

- Error rate of the linear classifier by linear programming (L2). This measure is the error rate of the linear classifier defined for L1, measured using the training set. Let $h(x)$ denote the linear classifier obtained. L2 is then given by:

$$L2 = \frac{\sum_{i=1}^{n} I(h(x_i) \neq y_i)}{n} \tag{2.7}$$

  Higher L2 values denote more errors and thus a greater complexity regarding the aspect that the data cannot be separated linearly. For the problem in Figure 2.4, the L2 value is $\frac{2}{33}$. L2 has similar issues to L1 in that it does not differentiate between problems that are barely linearly separable (i.e., with a narrow margin) and those with classes that are very far apart.

- Fraction of points on the class boundary (N1). This measure constructs a class-blind minimum spanning tree over the entire dataset, as illustrated in Figure 2.5. Herewith, each vertex corresponds to an example and the edges are weighted according to the distances between them. N1 is obtained by computing the percentage of vertices incident to edges connecting examples of opposite classes in the generated minimum spanning tree. These examples are either on the border or in the overlapping areas between the classes. Therefore, N1 estimates the size and complexity of the required decision boundary through the identification of the critical points in the dataset: those very close to each other that have opposite classes.

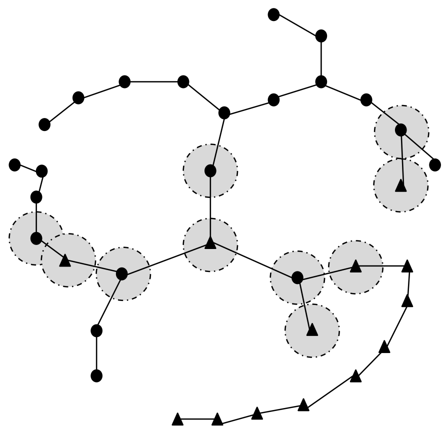


Figure 2.5: Example of minimum spanning tree generated for the problem for the Figure 2.2 and the detected points in the decision boundary.

Higher N1 values indicate the need for more complex boundaries to separate the classes and/or that there is a large amount of overlapping between the classes. However, N1 can be large even for a linearly separable problem when the distances between borderline examples are smaller than the distances between examples from the same class. Differently, Ho and Basu [60] suggested that a problem with a difficult nonlinear class boundary can still have relatively few edges among examples from different classes as long as the data points are compact within each class.

- Ratio of average intra/inter class NN distance (N2). This measure starts by computing the Euclidean distance from each data point to its nearest neighbor from the same class and its nearest neighbor from the other class. It then calculates the ratio between the average (over all points) of all distances to intraclass nearest neighbors and the average of all the distances to interclass nearest neighbors.

High values indicate that examples from the same class are disperse. N2 is sensitive to how data is distributed within classes and not only to how the boundary between the classes is like. It can also be sensitive to labeling noise in the data, just like N1.

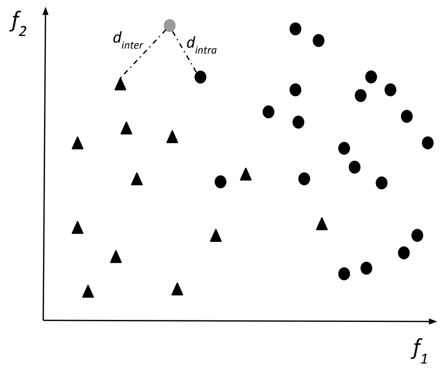


Figure 2.6: Example of intra and inter class distances for a particular example.

- Error rate of the 1-NN classifier (N3). This measure reports the error rate for a 1-NN classifier estimated by leave-one-out cross-validation on the training set. High values indicate that many examples are close to examples of other classes, making the problem more complex.

2.2.2. Measures of geometry, topology and density of manifolds

These measures indirectly characterize class separability by assuming that a class is made up of single and multiple manifolds that support the probability distribution of a given class.

- Nonlinearity of a linear classifier by linear programming (L3). This measure makes use of the linear classifier defined for L1 to report the error rate obtained on a test set created from the training set by linear interpolation between randomly drawn pairs of points from the same class. Herewith, two examples from the same class are chosen randomly and they are linearly interpolated (with random coefficients), producing a new example. Figure 2.7 illustrates the generation of six new samples (in gray) from a base training dataset. Then, a linear classifier is trained on the original data and its error rate measured in the new data points. This index is sensitive to how the data from a class is distributed in the border regions and also on how much the convex hulls which delimit the classes overlap. Higher values indicate a greater complexity. Letting $h_T(x)$ denotes the linear classifier induced from the original training data T, the L3 measure can be expressed by:

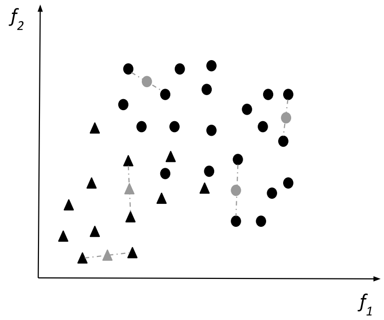


Figure 2.7: Example of how new points are generated in measures L3 and N4.

$$L3 = \frac{1}{l} \sum_{i=1}^{l} I(h_T(x'_i) \neq y'_i), \tag{2.8}$$

where l is the number of interpolated examples x'_i and their corresponding labels are denoted by y'_i. In this work, we generate the interpolated examples maintaining the proportion of examples per class from the original dataset and use $l = n$.

- Nonlinearity of the 1-NN classifier (N4). This measure creates a test set as proposed by L3 and returns the test error for the 1-NN classifier. High N4 values indicate problems of great complexity. In contrast to L3, N4 can be applied directly to multiclass problems, without the need to decompose them into binary subproblems first.

- Fraction of maximum covering spheres (T1). This measure counts the number of spheres needed to cover each class, where each sphere is centered at a training point and grown to the maximum size before it touches another class.

The count is then normalized by the total number of points. In a problem where each point is closer to points in the other class than to points in its own class, each point is covered by a distinctive sphere of a small size, resulting in a high value for the measure.

- Average number of points per dimension (T2). This is the simple ratio between the number of points in the dataset and the number of feature dimensions.

2.3. DNA Microarray

All cells have a nucleus, and inside this 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, with which it is possible to measure the simultaneous expression of tens of thousands of genes [99]. Figure 2.8 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 diseased states.

During the last two decades, the advent of microarray datasets has stimulated a new line of research both in bioinformatics and in machine learning. Although there are usually very small samples (often less than 100 patients) for training and testing, the number of features in the raw data ranges from 2000 to 25,000. A typical classification task is to separate healthy patients from cancer patients based on their gene expression "profile" (binary approach). There are also datasets in which the goal is to distinguish among different types of tumours (multiclass aproach), making the task even more complicated. Therefore, microarray data poses a serious challenge for machine learning researchers. Having so many fields relative to so few samples created a high likelihood of finding "false positives" due to chance (both in finding relevant genes and in building predictive models). Thus, it becomes necessary to find robust methods to validate the models and assess their likelihood.

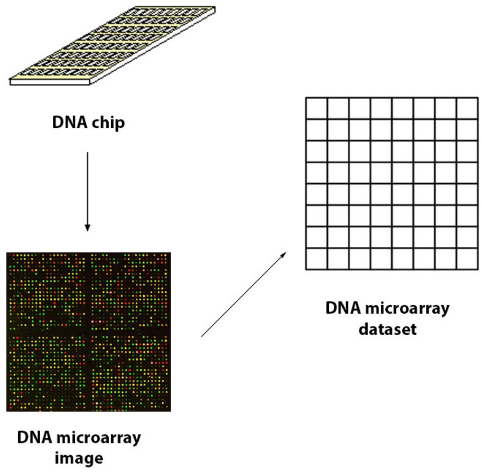


Figure 2.8: General process of acquiring the gene expression data from DNA microarray

Several studies have shown that most genes measured in a DNA microarray experiment are not relevant in the accurate classification of different classes of the problem [48]. To avoid the problem of the "curse of dimensionality" [66], feature (gene) selection plays a crucial role in DNA microarray analysis, which is defined as the process of identifying and removing irrelevant features from the training data, so that the learning algorithm focuses only on those aspects of the training data useful for analysis and future prediction [53].

Apart from the large number of genes versus small sample size, microarray data have other particularities such as the imbalance of the data, their complexity, the presence of overlapping, or the so-called dataset shift. These problematics render the analysis of microarray data an exciting domain [106].

- **Small sample size.** The first problem that one may find when dealing with microarray data is related to the small sample size (usually less than 100), as microarrays are still costly, although the price has been diminishing progressively. Thus, microarrays have increased the rate of data collection during the last years, but sample size is still a major issue when selecting features and building predictive models for medical applications. A key point in this regard is that error estimation is greatly impacted by small samples [41], as well as the low power for statistical tests, as for example the widely used t-test for comparison of groups [128].

As said above, the number of datasets available has been growing during the last years, although the sample size of each dataset remains small regarding the number of features. One way of dealing with this problem was to combine multiple datasets [91], that has the potential for increasing the power of microarray data analysis by pooling information. Combining datasets is however difficult, as different normalization and summarization techniques are used. Also, due to the shortage of datasets measuring the same area, perhaps different platforms might be needed, complicating even more the problem. That is perhaps the main reason behind the fact that most studies are limited to single datasets of small size. But combination might also be achieved by using fold-change methods, as feature selection is improved that way, and feature selection is almost mandatory for this type of datasets.

Without the appropriate estimation of the error, an unsound application of classification methods follows, which has generated a large number of publications and an equally large amount of unsubstantiated scientific hypotheses [23].To overcome this problem, it becomes necessary to select a correct validation method for estimating the classification error.

- **Class imbalance.** A common problem in real datasets is the so-called “class-imbalance problem”. Imbalance typically occurs when a dataset is dominated by a major class or classes with significantly more instances that the other classes in the data [59, 76]. In these cases, standard classification algorithms have a bias toward the classes with a greater number of instances, since the 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. This bias is even larger when data is high dimensional, such as microarrays: the high dimensionality further increases the bias toward the classification into the majority class, even if there is no real difference between the classes [79]. Therefore, minority class instances are more often misclassified than those from the other classes [46].

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, although in the global population there are more healthy patients than with cancer, among the population that undergoes certain clinical tests, there are more cancer patients than healthy. This results in multiple imbalanced tumor types or several skewed subtypes of a special tumor. In any case, it is important for practitioners to predict and prevent the appearance of rare/minority classes. Examples of very unbalanced microarray datasets are Lung-test or Brain-Tumor-1, among other (Figure 2.9). This problematic is of especial importance when the imbalance is more marked in the test set than in the training set. Multiclass datasets also suffer from this problem: some types of tumors/tissues have fewer samples compared to others. For example, Brain-Tumor-1 has five classes but the majority class takes 67% of the samples.

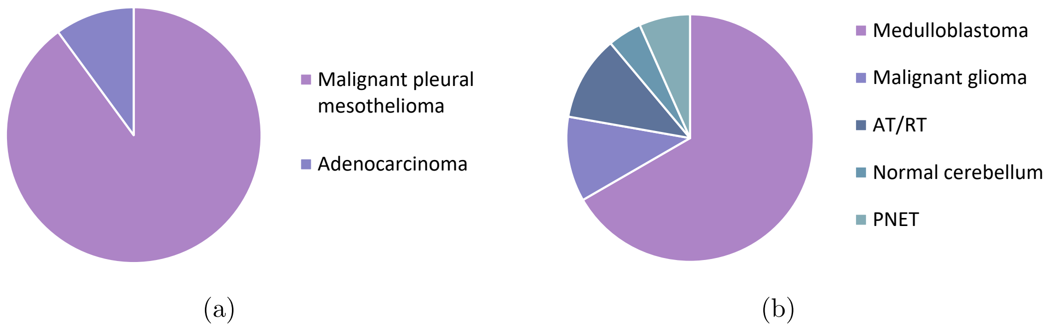


Figure 2.9: Class distributions for the (a) Lung-test and (b) Brain-Tumor-1 datasets

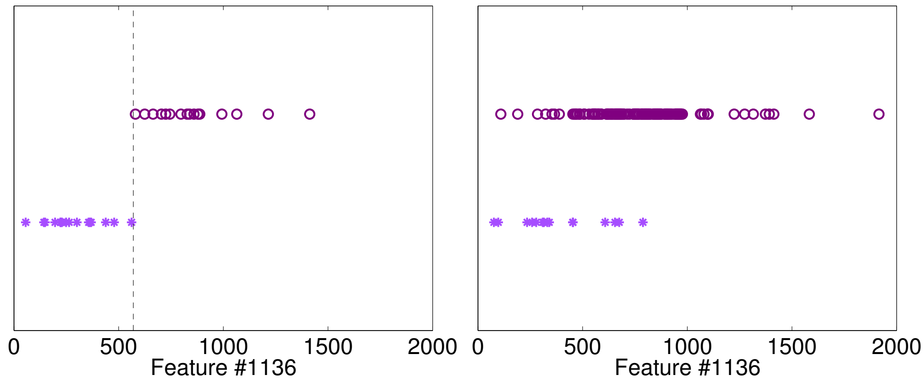
The traditional preprocessing techniques used to overcome this issue are undesampling and oversampling methods. Undersampling is a technique which creates a subset of the original dataset by eliminating samples. It aims to attain the same number of samples of the majority class as the minority class. As a very small number of samples are available in the microarray datasets, elimination of observations is not a good option. In contrast, oversampling methods create a superset of the original dataset by replicating some instances or creating new instances from existing ones. One of the most employed oversampling techniques is the so-called SMOTE [27], in which the minority class is oversampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. This technique was applied in Blagus and Lusa [80] on microarray data, although the authors stated that it does not attenuate the bias toward classification in the majority class for most classifiers. In recent years, ensemble of classifiers has arisen as a possible solution to the class-imbalance problem, attracting great interest among researchers [47], in several cases combined with preprocessing techniques such as SMOTE. Another approach to deal with imbalanced microarray datasets could be one-class SVM trained only with the target class, which may lead to a better predictive performance [82].

- **Dataset shift.** Another common problem when datasets were originally divided into training and test sets, is the so-called "dataset shift". This occurs when the testing (unseen) data experiences a phenomenon that leads to a change in the distribution of a single feature, a combination of features, or the class boundaries [87]. 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 and Prostate datasets have separated training and test sets. In this case, there is a single feature (#1136) which can correctly classify all the samples in the training set, as shown in Figure 2.10(a), in which 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 2.10(b). Furthermore, note that there is an enormous disparity in class distribution: 50%-50% in the training set and 90%-10% in the test set.

2.4. Experimental settings

Below we describe the specific datasets used for this study, focusing on some of their particularities, and we also introduce the classification algorithms and feature selection methods used, given the high dimensionality of microarray data.

Datasets. Two types of microarray datasets are described in the literature. The most famous are binary datasets that separate, for example, healthy patients from



(a) Lung Train (b) Lung Test

Figure 2.10: Feature #1136 in Lung dataset

patients with cancer. Multiclass datasets are more complicated datasets used to distinguish between different types of tumors. We chose 21 datasets: 11 binary and 10 multiclass. Two different situations are covered for our binary study: datasets divided into training and test sets and datasets with only a training set. Table 2.1 profiles the main characteristics of the microarray datasets used in this research in terms of the number of features, the number of samples and the imbalance ratio (IR), defined as the number of samples in the majority class divided by the number of samples in the minority class, where a high score indicates a highly imbalanced dataset.

As can be observed in Table 2.1, the datasets reflect some of the challenges mentioned above. Sample sizes are small, ranging from 21 to 253, whilst features number between 2000 and 24481. Dataset shift reflects the enormous complexity of the microarray data, with, in some cases, training and test samples recorded in completely different situations: for instance, the training samples for the Leukemia dataset were extracted from adult patients, whereas the test samples were obtained mainly from children; for the Prostate dataset, an independent set of testing samples was prepared from a different experiment, resulting in nearly 10-fold difference in overall intensity from the training data. Other datasets suffer from class imbalance, especially Lung test and Brain Tumor 1 (see Figure 2.9).

Feature selection methods. In the extreme case of having few observations for so many features, it is common to investigate the impact of feature selection. Fea-

Table 2.1: Characteristics of 21 microarray datasets.

Dataset	#Classes	#Features	#Samples	IR	Download
9-Tumors	9	5726	60	4.50	[115]
11-Tumors	11	12533	174	4.50	[115]
Brain	2	12625	21	2.00	[64]
Brain-Tumor-1	5	5920	90	15.00	[115]
Brain-Tumor-2	4	10,367	50	2.14	[115]
Breast-train	2	24,481	78	1.29	[2]
Breast-test	2	24,481	19	1.71	[2]
CLL-SUB-111	3	11,340	111	4.63	[122]
CNS	2	7129	60	1.86	[2]
Colon	2	2000	62	1.82	[2]
DLBCL	2	4026	47	1.04	[2]
Gli85	2	22,283	85	2.27	[122]
Leukemia-1	3	5327	72	4.22	[115]
Leukemia-2	3	11,225	72	1.40	[115]
Leukemia-train	2	7129	38	2.45	[2]
Leukemia-test	2	7129	34	1.43	[2]
Lung-cancer	5	12,600	203	23.16	[115]
Lung-train	2	12,533	32	1.00	[2]
Lung-test	2	12,533	149	8.93	[2]
Ovarian	2	15,154	253	1.78	[2]
Prostate train	2	12,600	102	1.04	[2]
Prostate test	2	12,600	34	2.78	[2]
Smk	2	19,993	187	1.08	[122]
SRBCT	4	2308	83	2.63	[115]
TOX-171	4	5748	171	1.15	[122]

ture selection methods have received a great deal of attention in the classification literature [19], which largely reflects *filter*, *wrapper* and *embedded* methods. Filter methods are based on performance evaluation metric calculated directly from the data, without direct feedback from predictors that will finally be used on data with reduced number of features. Wrappers involve a learning algorithm as a black box and consists of using its prediction performance to assess the relative usefulness of subsets of variables. Finally, embedded methods perform feature selection in the process of training and are usually specific to given learning machines. Since wrapper and embedded method interactions with the classifier are computationally burdensome, we opted for filter methods as more suitable for our microarray problem with a large number of features, as they have the key advantage of being less computationally intractable. For this purpose, two filters were chosen: CFS and

CONS; a description of both methods can be found in Appendix A.

Classifiers. Finally, four different classifiers, each belonging to a different family, were chosen to evaluate dataset complexity: two linear, naive Bayes (NB) and SVM using a linear kernel, and two non-linear, C4.5 and k-NN (see Appendix A). All four classifiers were executed employing the Weka tool [54], using default values for the parameters.

2.5. Experimental results

The aim of the experiments described below is to analyze the relationship between dataset complexity and the results obtained by the four classifiers described above. Four of the 21 datasets (Breast, Leukemia, Lung and Prostate) are divided into training and test datasets. For the remaining 17 datasets only training sets are available; since standard 10-fold cross-validation would reduce the test set in some of these latter datasets to just a handful of samples, we computed 5-fold cross-validation three times to estimate the error rate.

In order to cover all the cases, following the experimental framework described below we conducted two kinds of experiments, that is, for the 11 binary datasets and for the 10 multiclass datasets:

1. The complexity of the microarray datasets is analyzed so as to determinate the relationship between dataset complexity and the results obtained by the different classifiers.

2. Features are selected with a view to dimensionality reduction and the filters are applied.

3. The complexity of the new reduced datasets is analyzed for differences in complexity measures with the original datasets.

4. Statistical t-tests are performed to demonstrate that the results obtained are statistically significant.

2.5.1. Binary approach

In order to gain deeper insights into the 11 binary datasets studied, we subdivided the experiment into an analysis of datasets for which training and test sets are available and an analysis of datasets for which only training sets are available.

2.5.1.1. Is dataset complexity related to classifier accuracy?

Table 2.2 shows the results for the data complexity measures applied to the microarray datasets. Below we summarize results common to almost all the datasets, leaving more specific results to be commented later in detail.

- The fact that F2 = 0 for all the datasets indicated the presence of at least one feature that could discriminate between samples from the two classes.
- For all datasets T1 = 1. When a data point is closer to a point in the other class rather than in its own class, each point is covered by a small, distinctive sphere, resulting in a high T1 value.
- T2 always took extremely low values, since we are dealing with datasets in which the number of features is much higher than the number of samples.
- L2 = 0 for all the datasets excepting Colon, indicating this to be the only dataset in which classes were not linearly separable. By way of explanation, note the high L1 value for this dataset and, moreover, that this was the only dataset for which L2 was distinct from zero. Due to its small size, overfitting might have occurred during the training process, severely underestimating the true error rate.

Table 2.3 shows classification accuracy rates for the binary datasets (highest accuracy rates highlighted in bold). The area under the curve (AUC) is also shown, presenting analogous results as those for accuracy. For the Breast, Leukemia, Lung and Prostate datasets, results are shown for both training (Tr) and test (Te) sets. Regarding the training-set-only datasets, SVM performed best for all the datasets except Brain and Colon. In this study our choice of SVM with a linear kernel seems

Table 2.2: Complexity measures for 11 binary datasets.

	Overlapping			Class separability					Geometry, topology and density			
	F1	F2	F3	L1	L2	N1	N2	N3	L3	N4	T1	T2
Brain	0.894	0.000	0.905	0.001	0.000	0.762	0.989	0.619	0.000	0.000	1.000	0.002
Breast train	0.677	0.000	0.282	0.112	0.000	0.577	0.976	0.410	0.000	0.013	1.000	0.003
Breast test	4.981	0.000	0.895	0.018	0.000	0.789	1.002	0.684	0.000	0.000	1.000	0.001
CNS	0.448	0.000	0.333	0.023	0.000	0.567	0.993	0.417	0.000	0.042	1.000	0.008
Colon	1.083	0.000	0.419	0.498	0.032	0.322	0.891	0.242	0.016	0.064	1.000	0.031
DLBCL	2.907	0.000	0.702	0.141	0.000	0.362	0.929	0.234	0.000	0.012	1.000	0.012
Gli85	2.349	0.000	0.565	0.087	0.000	0.223	0.928	0.118	0.000	0.000	1.000	0.009
Leukemia train	4.303	0.000	0.921	0.038	0.000	0.237	0.914	0.105	0.000	0.000	1.000	0.005
Leukemia test	3.296	0.000	0.912	0.021	0.000	0.382	0.926	0.235	0.000	0.000	1.000	0.005
Lung train	6.568	0.000	0.937	0.011	0.000	0.062	0.912	0.000	0.000	0.000	1.000	0.003
Lung test	4.848	0.000	0.966	0.114	0.000	0.094	0.796	0.054	0.000	0.000	1.000	0.012
Ovarian	7.002	0.000	0.731	0.467	0.000	0.126	0.763	0.047	0.000	0.022	1.000	0.017
Prostate train	2.047	0.000	0.647	0.082	0.000	0.304	0.942	0.167	0.000	0.059	1.000	0.008
Prostate test	11.351	0.000	0.941	0.079	0.000	0.206	0.894	0.088	0.000	0.000	1.000	0.003
Smk	0.414	0.000	0.128	0.054	0.000	0.524	0.961	0.358	0.000	0.115	1.000	0.009

appropriate as, focusing on the results for L1 and L2, Colon appears to be the only dataset that was not linearly separable.

Regarding F1 measure, which indicates dataset complexity, Table 2.2 shows that the four datasets with the lowest complexity values (Brain, CNS, Colon and Smk) had the poorest classification accuracies (all below 80%). We would expect the N2 value to be high if samples from the same class were scattered over the finite space, complicating the classification task based on distances (the case with k-NN). Brain, CNS and Smk datasets obtained a poor k-NN performance but showed a high N2 value.

Datasets originally divided into training and test sets presented an added challenge for the ML methods, given that the data were extracted under different conditions. If the two sets were joined in a unique dataset (e.g., in order to later apply k-fold cross-validation), training would be easier and dataset shift of the microarray data would be overlooked.

- Accuracies for the training sets would be expected to be notably higher than for the test sets and might even show the effect of overfitting due to the small sample size in some cases. However, this only happened, in fact, for two datasets: Breast and Prostate (see Table 2.3). The data in Table 2.2 explains why. F1 values for the Breast and Prostate training and test sets differed significantly, whereas they were more similar and also higher for the Lung and Leukemia training and test sets (indicating simpler classification).

- Focusing on linearity, the L1 values for these datasets suggest that all were linearly separable, so we would expect good behavior for the linear classifiers (NB and SVM). Interestingly, the SVM classifier did not exhibit the same superior classification power with this type of datasets as it did for the training-set-only datasets. This may be because this method is sensitive to overfitting, most especially for independent training and test sets. In fact, SVM achieved 100% classification accuracy for all the four dataset training sets, with performance decays for the test sets (except for Leukemia).

This kind of analysis can be used to support the choice of the most promising classification algorithm. Figure 2.11 depicts the behavior of the four classifiers used

Table 2.3: Average classification accuracy and AUC for 11 binary datasets

		Brain	CNS	Colon	DLBCL	Gli85	Ovarian	Smk	Breast Tr/Te	Leukemia Tr/Te	Lung Tr/Te	Prostate Tr/Te
C4.5	Acc.	**0.92**	0.63	**0.79**	0.74	0.75	0.96	0.63	0.99/**0.74**	1.00/**0.91**	1.00/0.82	0.98/0.26
	AUC	0.89	0.65	0.85	0.75	0.71	0.95	0.65	0.99/0.79	1.00/0.91	1.00/0.78	0.99/0.50
NB	Acc.	0.67	0.62	0.63	0.91	0.81	0.92	0.62	0.59/0.37	1.00/0.88	1.00/0.96	0.64/0.26
	AUC	0.50	0.59	0.75	0.97	0.78	0.94	0.61	0.52/0.50	1.00/0.86	1.00/0.97	0.63/0.61
1-NN	Acc.	0.47	0.58	0.71	0.74	0.85	0.92	0.59	1.00/0.68	1.00/0.71	1.00/0.98	1.00/**0.53**
	AUC	0.46	0.58	0.74	0.75	0.85	0.91	0.65	1.00/0.72	1.00/0.66	1.00/0.95	1.00/0.64
SVM	Acc.	0.67	**0.67**	0.77	**0.94**	**0.92**	**0.99**	**0.71**	1.00/0.58	1.00/0.85	1.00/**0.99**	1.00/**0.53**
	AUC	0.50	0.62	0.79	0.91	0.89	0.97	0.76	1.00/0.63	1.00/0.82	1.00/0.99	1.00/0.68

in this study (C4.5, NB, 1-NN and SVM) for the domain of values of several data complexity measures and considering as input the seven training-only microarray datasets. We have excluded from this study the complexity measures (L2, L3, N3 and N4) that return error rates for certain classifiers and also complexity measures (F2, T1 and T2) with similar values for all datasets.

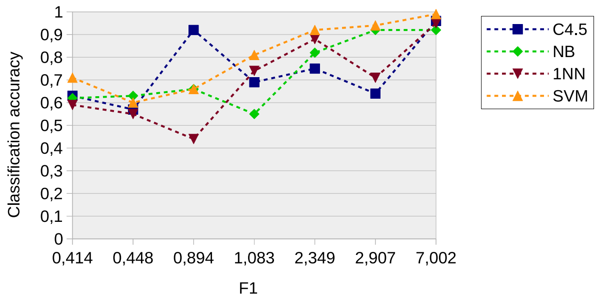
- The results for the F1 measure are shown in Figure 2.11(a). As the measure increases in value, overlapping between classes is reduced so better classification results can be expected. In terms of classification accuracy, all four algorithms behaved similarly, but more robust results are obtained by SVM classifier.

- Behavior for F3 (the other measure related to overlapping between classes) is similar (see Figure 2.11(b)). For all the classifiers, when feature efficiency was low the accuracy rate was negatively affected.

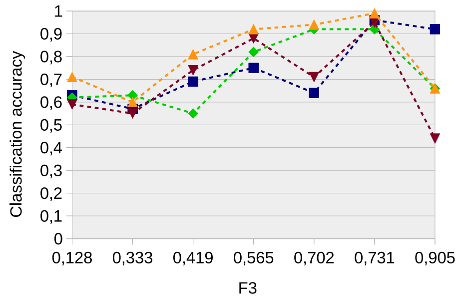
- Regarding the L1 measure (Figure 2.11(c)), small values should indicate high linear separability. However, as can be seen, this does not seem to occur and, as explained above, this might be because the measure can be sensitive to overfitting. In fact, the results for L1 after applying the CFS filter (see Figure 2.11(d))—feature selection reduces the overfitting risk—show that linear separability affected the accuracy rates of the classification algorithms, with the most robust results obtained for the SVM classifier.

- The results for N1 are shown in Figure 2.11(e). Higher values indicate that most of the points lie close to the class boundary, making it more difficult to define this class boundary accurately. Thus, the nature of this measure in terms of its NN definition would indicate N1 to be of interest in supporting the choice or otherwise of a 1-NN classifier.

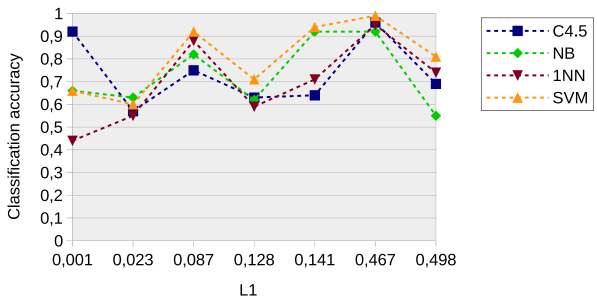
- The results for N2 are shown in Figure 2.11(f). High values, which is the case here, indicate that samples of the same class are disperse. Even so, N2, even though it is not as robust as N1, seems also to be of interest in supporting the choice or otherwise of a 1-NN classifier.



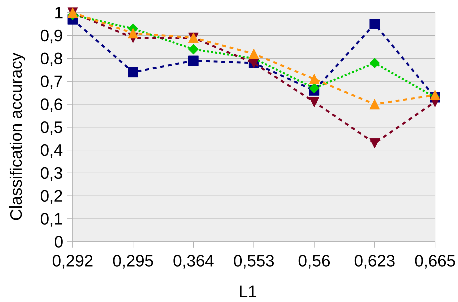
(a) Accuracy vs. F1 measure



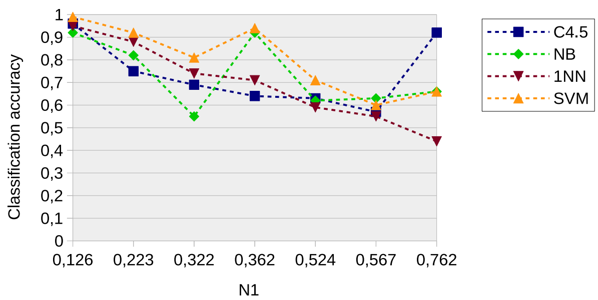
(b) Accuracy vs. F3 measure



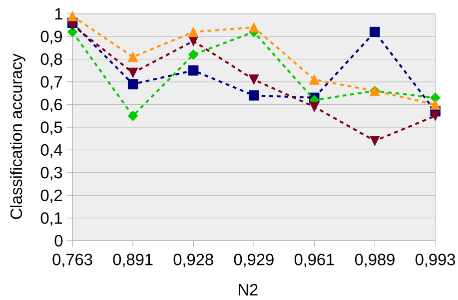
(c) Accuracy vs. L1 measure



(d) Accuracy vs. L1 measure after applying the CFS filter



(e) Accuracy vs. N1 measure



(f) Accuracy vs. N2 measure

Figure 2.11: Behavior of four classifiers for the domain of values of several data complexity measures.

2.5.1.2. Can feature selection reduce complexity?

Since several studies have shown that most genes measured for microarray purposes are not crucial to accurate classification, we applied the CFS and CONS filters in order to improve classification performance. Table 2.4 shows classification accuracies for the binary datasets after applying these filters, with the last row indicating the number of features (genes) required for each tested combination. The filters, but especially CONS, are able to considerably reduce the number of genes required to the order of dozens rather than thousands. Having far fewer genes to handle can facilitate identification of the underlying mechanisms relating gene expressions to particular diseases.

The CFS filter obtains the best accuracy rates for almost all the binary datasets and is especially powerful in its effect on the Prostate dataset, improving classification accuracy from 53% to 97%. The CONS filter, in contrast, does not work effectively with this type of data. In order to shed light on this issue, we applied the data complexity measures to the datasets after redundant and/or irrelevant genes were removed. Figure 2.12 illustrates some of these data complexity measures with and without feature selection.

- The N1 measure decreases considerably, especially for the CFS filter. Higher N1 values indicate smaller separation in distributions and a more difficult classification task, so it seems logical to think that lower values after applying the filters would improve classification performance. Nevertheless, that is not true for the CONS filter. This can be explained by the fact that F1 and F3 measures obtain lower values after applying CONS compared to CFS or to no feature selection, making the classification task more difficult. Furthermore, the fact that the CONS filter selected lower number of genes—perhaps too few—may be the reason why this filter failed to achieve results as good as those for CFS.
- The L1 and L2 values increase significantly after applying the CFS and CONS filters, which might be because feature selection reduces the risk of overfitting.
- N2 values for the original datasets are higher, suggesting that instances from the same class are dispersed in the feature space. However, values are lower

Table 2.4: Average classification accuracy and number of genes for 11 binary datasets.

		Brain	CNS	Colon	DLBCL	Gli85	Ovarian	Smk	Breast Tr/Te	Leukemia Tr/Te	Lung Tr/Te	Prostate Tr/Te
C4.5	CFS	**0.95**	0.63	0.78	0.74	0.79	0.97	0.66	1.00/**0.68**	1.00/0.91	1.00/0.82	0.99/0.26
	CONS	0.79	0.54	0.73	0.74	0.76	0.98	0.62	0.99/**0.68**	1.00/0.91	1.00/0.82	0.99/0.23
NB	CFS	0.78	0.63	0.80	**0.93**	0.84	0.99	0.67	0.60/0.37	1.00/**0.94**	1.00/**1.00**	0.94/0.26
	CONS	0.52	0.53	0.68	0.81	0.81	0.98	0.64	0.58/0.37	0.97/0.91	0.96/0.86	0.84/0.32
1-NN	CFS	0.43	0.61	0.78	0.89	**0.89**	**1.00**	0.61	0.97/**0.68**	1.00/0.88	1.00/0.99	0.98/**0.97**
	CONS	0.82	0.61	0.69	0.79	0.77	0.99	0.61	1.00/0.47	1.00/0.91	1.00/0.82	1.0/0.26
SVM	CFS	0.60	**0.64**	**0.82**	0.91	**0.89**	**1.00**	**0.71**	0.97/**0.68**	1.00 /0.88	1.00/0.99	0.98/**0.97**
	CONS	0.67	0.63	0.66	0.79	0.81	0.99	0.65	0.56/0.32	0.86/0.79	1.00/0.82	0.78/0.26
#genes	All	12625	7129	2000	4026	22283	15154	19993	24481	7129	12533	12600
	CFS	49.20	43.80	24.00	63.60	168.40	34.00	94.40	130	36	40	89
	CONS	1.00	3.80	3.40	2.60	2.80	3.00	9.8	5	1	2	4

after feature selection was applied, indicating a simplification of the data structure.

2.5.2. Multiclass approach

The same experimental framework is used for multiclass datasets. As most of the data complexity measures are not directly applicable to a multiclass problem, we use two well-known strategies to convert a multiclass problem to many instances of a two-class problems: *one-versus-one* (OVO) and *one-versus-rest* (OVR). OVO considers each binary pair of classes and OVR approach takes one class as positive and rest all as negative.

Although the simplest way to classify a dataset with more than two classes is to use a multiclass classifier, this approach runs the risk of focusing on the majority classes, so good results cannot be expected [45]. In order to overcome this problem, the multiple class problem is reduced to a series of binary problems that are solved individually and, the resulting predictions are combined to obtain a final solution. Although several class binarization techniques are proposed in the literature, we apply the widely used OVO strategy in combination with the Hamming decoding, loss-based decoding, cumulative sum and cumulative sum with threshold techniques [17] (only the best results are discussed in this study).

It was not possible to demonstrate that data complexity measures were correlated with classifier behavior, as the results were unclear. This is because in studying the complexity of a dataset with more than two classes, the values of the measures are related only to two classes (OVO strategy) or to one class versus the remaining classes (OVR strategy) whilst the classification performance refers to all the classes. However, in the following subsections we endeavor to identify some relationships.

2.5.2.1. Is dataset complexity related to classifier accuracy?

Table 2.5 briefly summarizes the results obtained by the different data complexity measures for the 10 multiclass datasets chosen following the OVR strategy. Indicated for each data property (overlap between classes, non-linearity, closeness to class boundary and sample dispersion with respect to the class groupings) are the corresponding classes and the related data complexity measures. For example, in the Brain Tumor 1 dataset, class 0 is not linearly separable from the whole class. When all classes present a given problematic, this fact will be indicated with an asterisk (*).

An alternative option could be to use the $1/mF1$ measure, described in Section 2.2, with simple problems represented by small values of this measure. Accordingly, the most difficult datasets to classify should be 9-Tumors and TOX-171 whilst 11-Tumors should be easy to classify. However, these assumptions were only true for 9-Tumors and 11-Tumors—which, respectively showed poor (54%) and good (91%) classification performance rates—but not for TOX-171, which, despite obtaining the highest value for the $1/mF1$ measure, obtained a 96% classification performance rate. This may have happened because this measure, in trying to summarize the complexity of all the classes at once, ignored the particular cases.

Table 2.5: Summary results for 10 multiclass datasets.

	Complexity						
	Overlap (F1, F3)	Non-linearity (L1)	Boundary Class (N1)	Disperse (N2)	$1/mF1$	IR	Max. Accuracy
CLL-SUB-111	1,2	1,2	1,2	*	0.19	4.63	0.76
Leukemia 1		0,2		*	0.23	4.22	0.96
Leukemia 2	1	0,1,2		*	0.26	1.40	0.96
Brain Tumor 2	2	0,2,3	2	*	0.14	2.14	0.71
SRBCT				*	0.26	2.63	1.00
TOX-171	*	*		*	0.64	1.15	0.96
Brain Tumor 1	0,4	0		*	0.16	15.00	0.96
Lung cancer	0	0		*	0.14	23.16	0.95
9-Tumors	0,1,2,3		1,5	*	0.30	4.50	0.54
11-Tumors	2,9			*	0.06	4.50	0.91

Table 2.6, which shows the complexity measures results for the CLL-SUB-111 dataset for both the OVO and OVR strategies, indicates that the classification algorithms had problems in distinguishing between samples from classes 2 and 3, which, in fact, obtained the lowest values for the F1 and F3 measures. These results are consistent with the high values obtained for N1, indicating that most points lay close to the class boundary and thereby making the classification task more difficult. As for L1 and L2, which suggest non-linearity of the data, it seems that class 1 was

the only class that was linearly separable. These results explain the true positive rates for each class, with class 0 obtaining higher rates than classes 1 and 2.

Table 2.6: Analysis of the CLL-SUB-111 dataset.

		Data complexity measures						% True positives			
		F1	F3	L1	N1	N2		C4.5	NB	1NN	SVM
OVO	0 vs. 1	15.953	0.966	0.093	0.033	0.770	Class 0	**75.33**	**92.44**	**94.67**	**94.67**
	0 vs. 2	16.134	0.967	0.089	0.032	0.750					
	1 vs. 2	0.871	0.230	0.001	0.610	0.992	Class 1	58.78	60.69	39.27	64.34
OVR	0 vs. rest	16.244	0.982	0.198	0.126	0.849					
	1 vs. rest	1.047	0.234	0.883	0.550	0.995	Class 2	58.11	71.00	68.17	84.48
	2 vs. rest	0.585	0.216	0.919	0.505	0.993					

2.5.2.2. Can feature selection reduce complexity?

Figure 2.13 illustrates the behavior of several data complexity measures with feature selection (CFS and CONS) and without feature selection (All genes), where the X axis represents the multiclass datasets used. The value of the complexity measure was the average of the OVR problems into which each multiclass dataset was divided. Standard deviations are also shown.

As happened with the binary datasets, complexity in some datasets was reduced after applying feature selection. The first conclusion is that the N1 and N2 measures decreased significantly, indicating a simplification of the data structure. The L1 measure increased considerably after applying feature selection (Figure 2.13(c)), possibly because feature selection avoided overfitting. In contrast, the F1 and F3 measures obtained lower values after the filters (especially CONS) were applied, which should reflect a more difficult classification task. As for the standard deviation values, it can be observed that these decreased for almost all the measures after feature selection was applied, indicating that class complexities were more homogeneous.

2.5.3. Are the results statically significant?

In this section we will study if the observed differences/correlations are significant or just the result of chance in this particular set of microarray datasets, as it is discussed in [22]. Taking this context into account, a paired t-test [37] was performed so as to compare classification performances. The aim was to demonstrate that the recommended classifiers obtained in fact better statistically significant results than those of the other classifiers. Two different contexts were analyzed, without using feature selection, and using a previous filter as preprocessing.

The considered classification algorithms were as above, C4.5, NB, kNN and SVM. Feature selection is performed by the CFS and CONS filters. The study is conducted for two different setups: for the seven binary datasets with only training set (Brain, CNS, Colon, DLBCL, Gli85, Ovarian and Smk) and for all the 21 microarray datasets chosen for this work (see Table 2.1).

Table 2.7 shows the results of the one-side paired t-test for the nine pairwise differences of classification accuracy rates. In the study based on the seven binary datasets, only three of the nine pairwise differences between accuracy rates are statistically significant when tested at an α-value of 0.05. As Boulesteix et al. concluded in [22], significant differences would not have been detected with the number of datasets being 3 to 7, especially in the context of high dimensional problems. However, if the study is extended to the 21 datasets, the nine pairwise differences between classification accuracy rates are statistically significant.

These results, besides highlighting the cruciality of using a sufficient number of datasets in comparison studies, show the superiority in performance of SVMs over other classifiers in this domain [90], both before and after (CFS filter) applying feature selection. In the case of the CONS filter, the results do not hold, and in this case the best classification performances are achieved by kNN.

2.6. Summary

We have analyzed the theoretical complexity of several datasets, trying to relate this to the results achieved by four widely-used classifiers. Making use of several data

Table 2.7: Results of the one-sided matched-pair t-test for the nine pairwise differences of accuracy rates over 7 and 21 datasets.

		Paired t-test		
Datasets	Comparison	$\overline{\Delta acc}$	t	p-value
7 binary datasets	C4.5 vs. SVM	-0.036	-0.638	0.273
	NB vs. SVM	-0.07	-3.862	**0.004**
	kNN vs. SVM	-0.116	-5.036	**0.001**
	C4.5-CFS vs. SVM-CFS	-0.007	-0.113	0.457
	NB-CFS vs. SVM-CFS	0.01	0.337	0.375
	kNN vs. SVM	-0.051	-2.184	**0.036**
	C4.5-CONS vs. SVM-CONS	-0.005	-0.203	0.422
	NB-CONS vs. SVM-CONS	-0.033	-1.322	0.117
	kNN-CONS vs. SVM-CONS	0.011	0.458	0.331
21 binary & multiclass datasets	C4.5 vs. SVM	-0.107	-3.402	**0.001**
	NB vs. SVM	-0.095	-4.914	**4.183e-005**
	kNN vs. SVM	-0.094	-5.732	**6.55e-006**
	C4.5-CFS vs. SVM-CFS	-0.123	-2.936	**0.004**
	NB-CFS vs. SVM-CFS	-0.068	-1.796	**0.044**
	kNN vs. SVM	-0.036	-3.627	**0.001**
	C4.5-CONS vs. SVM-CONS	0.047	1.814	**0.042**
	NB-CONS vs. SVM-CONS	0.046	2.453	**0.012**
	kNN-CONS vs. SVM-CONS	0.059	3.465	**0.001**

complexity measures, we studied in detail the problematics present in 21 microarray binary and multiclass datasets. Our main conclusions are the following:

- The F1 and F3 measures related to overlapping between classes have been demonstrated to be related to classification accuracy, with better classification results to be expected for increased values of these measures.

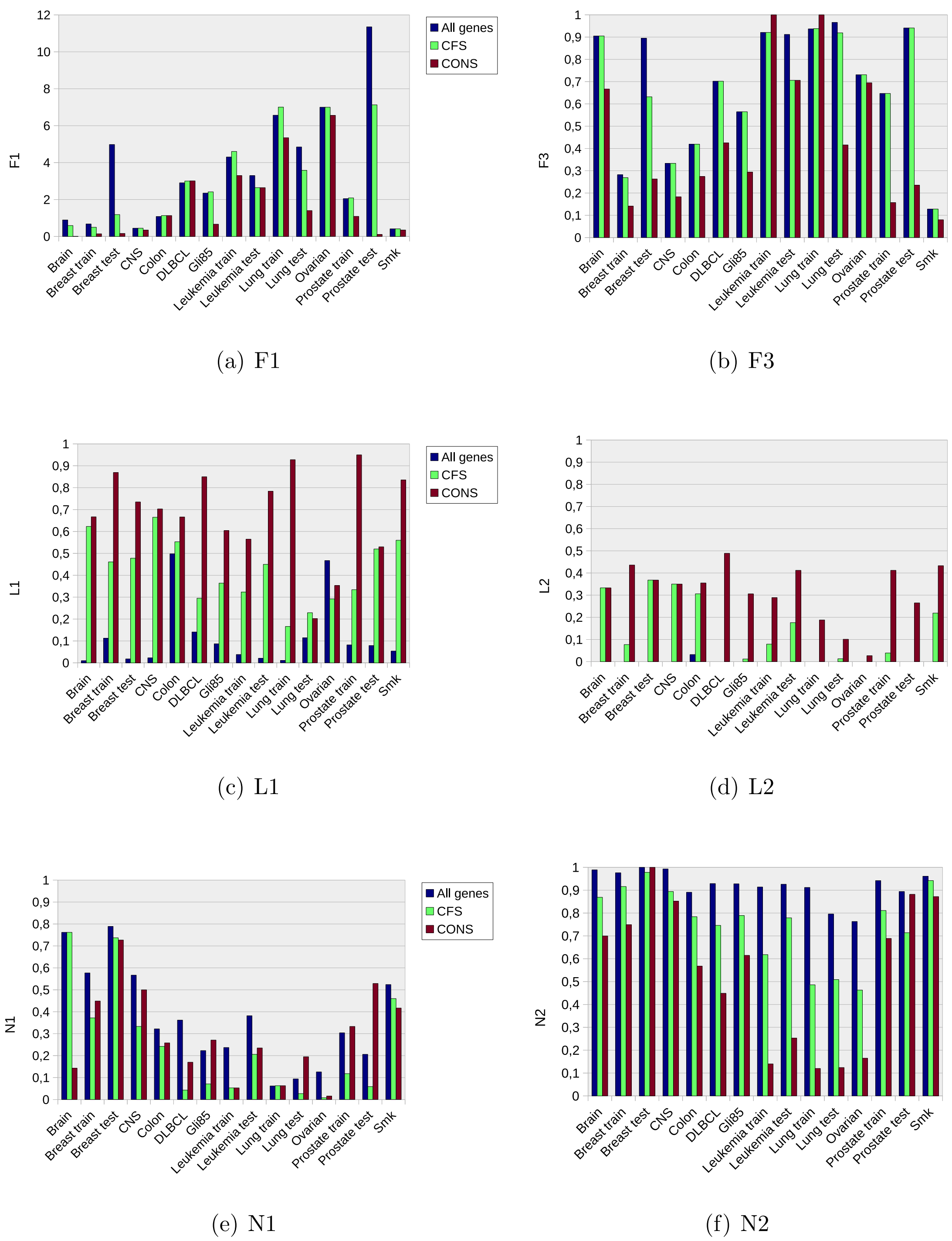
  F2—which is 0 for all datasets—does not yield any information. A side effect of employing the product (see Eq. 2.3) is that the value of the measure decreases greatly as dimensionality increases. That is, it is highly dependent on the number of features a dataset has. This worsens for problems with many features, such as DNA microarrays, so that their F2 values may not be comparable to those of other problems with few features. Thus, in future work, it might be interesting to compute the logarithm or the sum rather than the product. Nonetheless, the result will not be an overlapping volume, but the amount or size of the overlapping region.

- The N1 and N2 separability measures are useful in predicting the behavior of the 1-NN classifier, whilst L1 is observed to be more related to SVM. As for L2 and N3, these measures are not useful for analyzing classifier behavior as they return the error rate of a specific classification algorithm.

- Regarding the measures of geometry, topology and density of manifolds, the impact of the T1 and T2 measures on the classification task is not representative as values were similar for all the microarray datasets. The problem with L3 and N4 is the same as explained above for L2 and N3.

To sum up, we recommend using F1 and F3 in all cases, N1 and N2 to analyze datasets for k-NN and L1 for linear classifiers (in this case, SVM).

Once dataset theoretical complexity was analyzed, we applied feature (gene) selection using two filters to reduce the number of features. We have observed that feature selection—especially via the CFS filter—reduces data complexity, resulting in the best classification performance for 15 of 21 datasets in this study. Due to the statistical tests performed over the 21 microarray datasets, we can conclude that the presented results are statistically significant. Moreover, although researchers agree that the best classifier simply does not exist, in this specific work, SVM with linear kernel was a good option, in general, reporting high classification accuracies.



(a) F1 (b) F3

(c) L1 (d) L2

(e) N1 (f) N2

Figure 2.12: Data complexity measures with feature selection (CFS and CONS) and without feature selection (All genes).

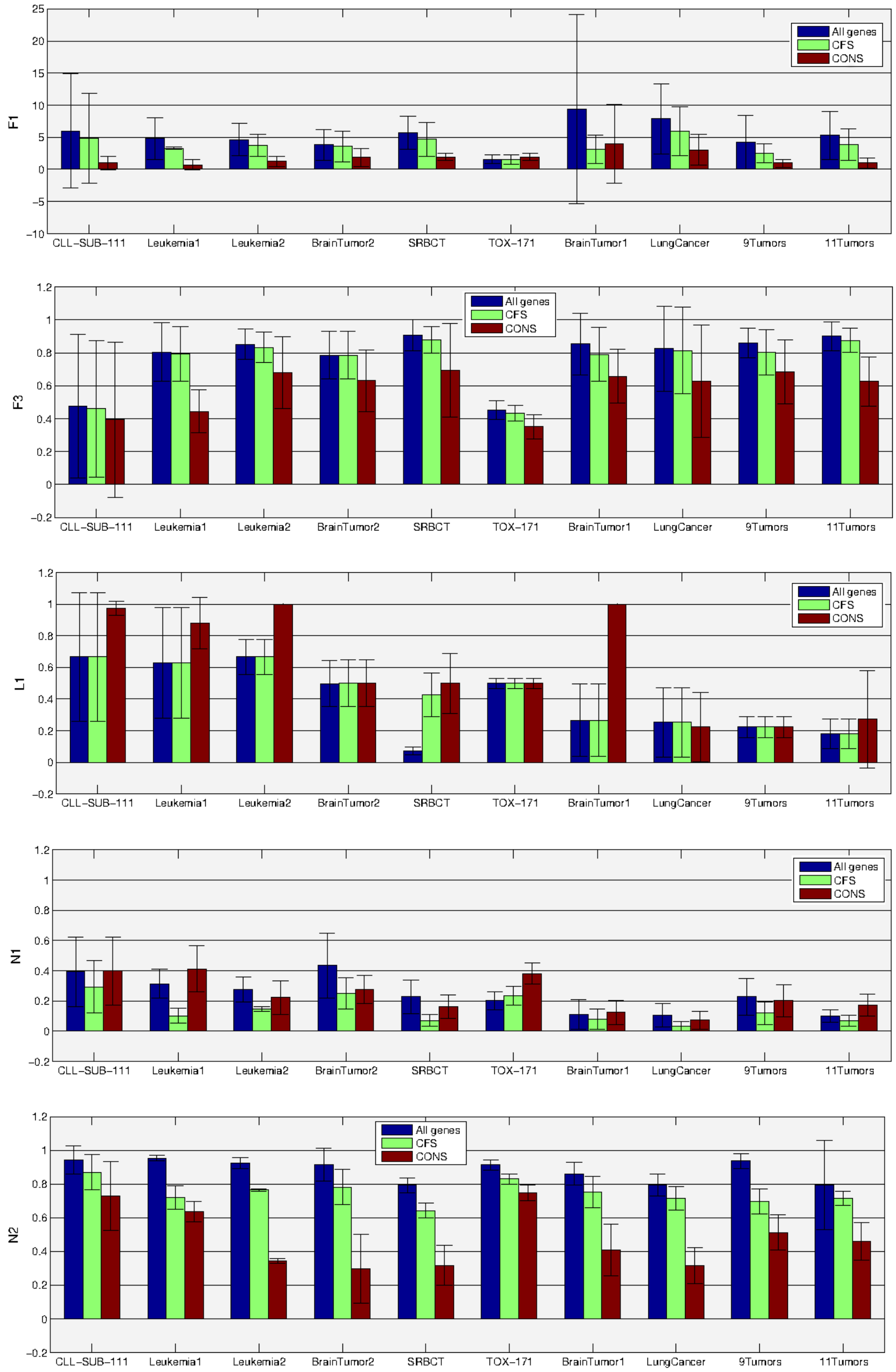


Figure 2.13: Complexity data measures with feature selection (CFS and CONS) and without feature selection (All genes) for 10 multiclass datasets.

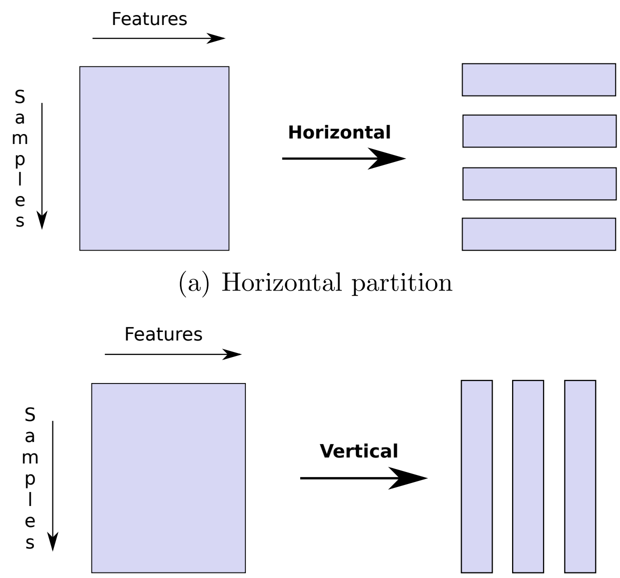
Chapter 3

Distributed learning

In the era of big data, with datasets rapidly growing in size—both in number of samples and features—, machine learning techniques face a major challenge since they cannot look at very large datasets and plausibly find a good solution with reasonable requirements of computation. The scaling up problem appears in any traditional machine learning algorithm when data size increases beyond capacity, damaging performance and efficiency. This issue can also affect negatively in some other aspects such as excessive storage requirements, increase of time complexity and, finally, generalization accuracy due to over-fitting and noise. Thus, the machine learning community has been essentially focused on the design of distributed or parallel algorithms to deal with massive datasets [98]. A method is said to be parallel when the processes are executed on different cores of one or several nodes connected by a high-speed network at the same location. Differently, a method is said to be distributed if the data are allocated in different locations and there is very low interaction among the processes. In this situation, the philosophy of distributed approach seems to be a promising line of research to reduce the current complexity of learmning algorithms. It represents a natural manner for scaling up algorithms since an increase of the amount of data can be compensated by an increase of the number of locations wherein the data is processed.

Data can be distributed either horizontally or vertically. In horizontal partitioning, the dataset is divided into several packets that have the same features as the original dataset, each containing a subset of the original instances (see Figure

3.1(a)). In vertical partitioning, the original dataset is divided into several packets that have the same number of instances as the original dataset, each containing a subset of the original set of features (see Figure 3.1(b)).



(a) Horizontal partition

(b) Vertical partition

Figure 3.1: Techniques to partition data.

This chapter is focused on horizontal partitioning, since it constitutes the most suitable and natural approach for most applications. In addition to the common learning scenario assumptions, we assume the availability of a test set large enough to obtain distributional information. In this distributed scenario, class probabilities can be shown to be a weighted average of the individual class probabilities within each node. These weights depend on the marginal probabilities of the instance over each node and over the entire dataset. This result motivates the study of the use of distribution distances for improving classification performance.

Besides, we assess a common problem in many real world problems, the "class-prior change" [42] in classification. In a non-distributed framework, this problem appears when the class balance changes between training and test datasets, due to sample selection bias or non-stationarity of the environment [101]. In our case, unbalancedness happens when the feature distributions differ between nodes. Distributed real-world datasets are usually not symmetric, i.e. the distributions of data for different locations may not be the same. Imagine a group of epidemiologists studying the spread of hepatitis-C in Europe. They are interested in detecting any underlying relation of the emergence of hepatitis-C in Europe with the weather and social patterns. They have access to some large hepatitis-C country-specific databases and an environmental database at EEA (European Environment Agency). These patterns could change considerably from one country to another (e.g. from Denmark to Italy). Moreover, analyzing the data from these distributed datasets using a traditional data mining algorithm will require combining the databases at a single location, which is quite impractical, or perhaps not possible due to memory reasons or to privacy issues.

3.1. Background

Different from the traditional centralized algorithms where a single learner has access to the full dataset, distributed learning algorithms have their foundations in ensemble learning [16]. Ensemble learning consists of a hierarchy of multiple local learners operating on subsets of the full dataset, and one or more ensemble learners combining the outputs of all the local learners. Thus, the ensemble approach is almost directly applicable to a distributed scenario since a classifier can be trained at each site, using the subset of data stored in it, and then the classifiers can be eventually combined using ensemble strategies. To combine the predictions of a set of classifiers, one of the simplest ways consists of using decision rules [69]. These decision/fixed rules are defined as functions that receive as inputs the outputs of the set of learned classifiers and combine them to produce a unique output.

During the last decade, several computation frameworks have emerged to ease the use of large quantities of data. MapReduce [36] is a simple model for distributed computing that abstracts away many of the difficulties in parallelizing data management operations across a cluster. Hadoop [5], built from commodity hardware and based on MapReduce processing, is a set of algorithms for distributed storage and processing of very large datasets on computer clusters. Apache Spark [6] is a fast, general engine for large-scale data processing, popular among machine learning researchers due to its suitability for iterative procedures. Developed as part of the Apache Spark project was MLlib [7], created as a scalable machine learning containing many algorithms and utilities. Another solution to the scability problem is the use of graphic processing units (GPUs) to distribute and thus accelerate calculations in learning algorithms.

3.2. A novel distributed learning model

Consider a population Ξ characterized by pairs of measurements $(x, C) \in \mathcal{X} \times \mathcal{C}$, where $\mathcal{X}$ denotes a domain of d-dimensional feature vectors and $\mathcal{C} = \{C_1, \ldots, C_m\}$ is a set of m class labels. Denote by $P(x, C)$ the joint probability function over $\mathcal{X} \times \mathcal{C}$. In a standard classification context, a classifier based on a training sample of n labeled objects $\mathcal{Z} = \{(x_1, C_1), \ldots, (x_n, C_n)\}$ is used to predict the class of unlabeled features. In a horizontally distributed framework, the population Ξ spreads across p disjoint nodes, let us say $\mathcal{P} \equiv \{\mathcal{N}_1, \ldots, \mathcal{N}_P\}$, and the training dataset brings together instances from the different nodes, i.e. $\mathcal{Z} = \cup_{i=1}^{p} \mathcal{Z}_i$, with $\mathcal{Z}_i$ formed by n_i instances belonging to $\mathcal{N}_i$, with $\sum_{i=1}^{p} n_i = n$.

In this paper we focus on a distributed environment where a set $\mathcal{T}$ of t unlabeled instances is available and our target is to estimate their labels. The availability of a whole set of unlabeled instances $\mathcal{T}$ enables us to gain knowledge about the underlying distribution of X and to assess how well this distribution is represented at each node. As we will see later, our learning model relies on these distributional distances so that the availability of $\mathcal{T}$ is a basic requirement.

In addition, our learning model is constructed under the standard assumption that the training set $\mathcal{Z}$ and the test set $\mathcal{T}$ are independent and identically distributed samples drawn from the population in study, and therefore following the probability model given by $P(x, C)$. Note that this stationarity assumption is the default assumption in many learning scenarios. No distributional or other assumptions are required on the data or how they are distributed across nodes. In fact, data within each node could follow different distributions since no constraints on the fragmentation scheme are imposed. In particular, our framework encompasses scenarios with unbalanced nodes [105] or with data-driven partitions on the basis of heuristic rules stated to obtain better classification rates [38].

We also impose the restriction that no communication between nodes is required. Intelligent interaction between nodes, e.g. taking advantage of the most informative data at each node, can improve the classification accuracy [33]. Nevertheless, exchanging information between nodes is frequently unfeasible in real problems dealing with distributed data for different reasons such as storage cost, communication cost or private and sensitive data, among others [120].

3.2.1. An overview of our approach

A common approach in distributed learning [12] consists of building classifiers trained at each node $\mathcal{N}_i$ using $\mathcal{Z}_i$, $i = 1, \ldots, p$, and then combining the classifier outputs by means of a proper ensemble learning strategy [39]. In our approach, we intend to take advantage of the availability of $\mathcal{T}$ to gain insight into the marginal probability distribution of the feature vectors, and using this knowledge to modulate the importance of each individual classifier in the combination rule. Specifically, we wish to estimate the posteriori probability that the j-th instance in $\mathcal{T}$, with observed feature vector x_j, belongs to the class C_k, for $k = 1, \ldots, m$ and $j = 1, \ldots, t$. Under the stationarity assumption and given that $\mathcal{P}$ is a partition of Ξ, we have

$$P\left(C_k \left| x_j \right.\right) P\left(x_j\right) = \sum_{i=1}^{p} P\left(C_k \left| x_j, \mathcal{N}_i \right.\right) P\left(x_j \left| \mathcal{N}_i \right.\right) P\left(\mathcal{N}_i\right), \tag{3.1}$$

where $P\left(x_j\right)$ denotes the marginal density of x_j, $P\left(x_j \left| \mathcal{N}_i \right.\right)$ the density of x_j conditional on the i-th node, and $P\left(\mathcal{N}_i\right)$ the prior probability of an instance is allocated to $\mathcal{N}_i$.

Let ω_{ji} be the ratio defined by $\omega_{ji} = \dfrac{P\left(x_j \left| \mathcal{N}_i \right.\right)}{P\left(x_j\right)}$, for $i = 1, \ldots, p$. Then, from (3.1) follows that

$$P\left(C_k \left| x_j \right.\right) = \sum_{i=1}^{p} P\left(C_k \left| x_j, \mathcal{N}_i \right.\right) \omega_{ji} P\left(\mathcal{N}_i\right). \tag{3.2}$$

Equation (3.2) establishes that the posteriori probability of the class C_k given an observed feature vector x_j is a weighted average of the posteriori probabilities within each node, with weights depending on the node size and the ratios ω_{ji}. By definition, ω_{ji} measures how well represented is the observed feature vector x_j in the i-th node. Whether the partition $\mathcal{P}$ has been set evenly and uniformly at random, the feature vectors within each node follow similar distributions and ω_{ji} will take values close to one for all j and i. Otherwise, markedly unbalanced nodes will produce very different ω_{ji}.

The value of $P\left(C_k\left|x_j\right.\right)$ can be directly estimated from (3.2) as long as the remaining involved probabilities are previously approximated. The posteriori probability within each node, $P\left(C_k\left|x_j, \mathcal{N}_i\right.\right)$, is estimated using the classifier trained at $\mathcal{N}_i$ and whose output consists of a vector of m belief values. The proportion of training data belonging to the i-th node can be taken as an estimate of $P\left(\mathcal{N}_i\right)$. For the sake of simplicity and computational efficiency, we will assume nodes of equal size so that the weight of a single prediction is not affected by the nodes' sizes. Lastly, the behavior in probability of the feature vectors over Ξ and over each node $\mathcal{N}_i$ can be modeled with nonparametric kernel densities based on the features forming $\mathcal{T}$ and $\mathcal{Z}_i$, respectively. Nevertheless, this involves several difficulties. First, we could face the "curse of dimensionality" problem since the dimension of the feature space may be arbitrarily large. Moreover, we look for a learning model able to manage different types of features, including mixtures of discrete and continuous variables. But even assuming an affordable dimension and continuous features, $(p+1)$ kernel densities should be obtained, which substantially increases the likelihood of estimation errors. In particular, small errors estimating $P\left(x_j\left|\mathcal{N}_i\right.\right)$ or $P\left(x_j\right)$ might produce arbitrarily large or small coefficients ω_{ji}, thus leading to overweight or underweight the predictions in specific nodes.

To overcome these drawbacks, the computation of the $(p+1)$ kernel densities is circumvented by directly estimating the coefficients ω_{ij}. Two different approaches are proposed. In both cases, the aim is to measure the dissimilarity d_i between the feature distributions on the i-th node and the global population, hereafter denoted by $F_{\mathcal{N}_i}$ and $F_{\mathcal{X}}$, respectively. A suitable distance between high-dimensional distributions is considered, and then specific values for d_i, $i = 1, \ldots, p$, are obtained using the empirical distributions based on $\mathcal{Z}_i$ and $\mathcal{T}$. The first proposal consists in taking $\omega_{ji} = K \cdot d_i^{-1}$, for all j and i, and K being a constant. This way, all the instances in $\mathcal{T}$ receive the same weight at each node, which decreases with the distance between $F_{\mathcal{N}_i}$ and $F_{\mathcal{X}}$. The second proposed approach is not simply based on the global distance between empirical distributions. The weights ω_{ji} are determined in order to maximize the matching between $F_{\mathcal{N}_i}(x_j)$ and $F_{\mathcal{X}}(x_j)$, for all $x_j \in \mathcal{T}$. Unlike the prior approach, the test instances receive different weights ω_{ji} at the same node. The effective computation of the weights is formalized throughout an optimization problem. A detailed description of the two proposed weighting criteria is provided in Section 3.2.4.

3.2.2. Outline of the methodology

We propose a distributed learning methodology consisting of the following four stages.

Step 1 Assess the distance between the probability distributions of X on the i-th training node $\mathcal{N}_i$ and the global population Ξ using a suitable statistic to measure dissimilarity between high-dimensional distributions. Denote by d_i the normalized distance obtained for the i-th training node, $i = 1, \ldots, p$.

Step 2 Based on a pre-selected classifier, obtain for each feature vector x_j of the test sample the classifier outputs $\mathbf{Y}_j = \{\mathbf{y}_{j1}, \ldots, \mathbf{y}_{jp}\}$, where $\mathbf{y}_{ji}$ denotes the response generated by the classifier trained at the node $\mathcal{N}_i$, for $i = 1, \ldots, p$ and $j = 1, \ldots, t$.

It is assumed that each classifier output consists of a vector of m membership or belief values, i.e. $\mathbf{y}_{ji} = (y_{ji1}, \ldots, y_{jim})$, where y_{jik} can be interpreted as the amount of confidence or evidence in the assignment of the feature x_j to the k-th class, C_k, for $k = 1, \ldots, m$.

Step 3 Obtain weighted versions of the belief values $\mathbf{Y}_j^\omega = \left\{\mathbf{y}_{j1}^\omega, \ldots, \mathbf{y}_{jp}^\omega\right\}$, with $\mathbf{y}_{ji}^\omega = \omega_{ji}\mathbf{y}_{ji}$, where the weights ω_{ji} take into consideration the distributional distances d_i. Two different criteria are proposed to determine how the weights are constructed.

Step 4 Generate a unique decision for classifying the j-th instance in $\mathcal{T}$, with observed feature x_j, by combining the corresponding weighted belief sets $\{\ \omega_{j1}\mathbf{y}_{j1} \ldots, \omega_{jp}\mathbf{y}_{jp}\ \}$. Following Kittler *et al.* [69], several fixed rules (*decision rules*) involving functions of the elements of $\mathbf{Y}_j^\omega$ are considered to produce the required unique output.

The key points of the proposed methodology involve: (i) the choice of the distributional distance d_i, (ii) the weighting criteria on the belief values regarding the distances d_i, and (iii) the selection of a decision rule. Each of these issues is properly discussed below.

3.2.3. Measuring dissimilarity between high dimensional distributions

To assess the distance between the probability distributions of X over an arbitrary node $\mathcal{N}_i$ and the population Ξ, we propose to use the so-called *energy statistic* [116, 117]. Consider two independent samples $\mathcal{X}$ and $\mathcal{X}'$ generated from multivariate distributions $F_{\mathcal{X}}$ and $F_{\mathcal{X}'}$, respectively. The energy distance between $\mathcal{X}$ and $\mathcal{X}'$ is defined by

$$E\left(\mathcal{X}, \mathcal{X}'\right) = 2d_{\mathcal{X},\mathcal{X}'} - d_{\mathcal{X},\mathcal{X}} - d_{\mathcal{X}',\mathcal{X}'}, \tag{3.3}$$

with

$$d_{\mathcal{A},\mathcal{B}} = \frac{1}{rs}\sum_{u=1}^{r}\sum_{v=1}^{s}\left\|a_u - b_v\right\|,$$

where $\|\cdot\|$ denotes the Euclidean norm and $\mathcal{A} \equiv (a_1, \ldots, a_r)$ and $\mathcal{B} \equiv (b_1, \ldots, b_s)$ denote arbitrary datasets.

Under mild regularity conditions on the generating patterns, Székely and Rizzo [116] established the consistency of the statistic (3.3) to check the equality of the generating distributions $F_{\mathcal{X}}$ and $F_{\mathcal{X}'}$. Hence $E\left(\mathcal{X}, \mathcal{X}'\right)$ can be seen as a measure of the distance between $F_{\mathcal{X}}$ and $F_{\mathcal{X}'}$ in such a way that the larger value of the statistic, the more distant are the distributions. By construction, $E\left(\mathcal{X}, \mathcal{X}'\right)$ is based on comparing averages of interpoint distances evaluated within and between samples, which means to move the multidimensional problem to dimension one. Thus, the energy distance is particularly attractive to be applied in arbitrarily high dimension. It is also worthy to remark that different types of interpoint distances could be used to construct $E(\cdot, \cdot)$, thus providing versatility to deal with features taking nominal, categorical, continuous and also mixed values. Also, the good analytical properties of the energy distance will allow us to formalize in the next section a suitable optimization problem designed to provide useful weights for the belief values. Supported by these nice properties, we decided to evaluate the distributional distance between each $\mathcal{N}_i$ and Ξ by means of the energy distance between the i-th training sample and the test sample, i.e. by $d_i = E\left(\mathcal{Z}_i, \mathcal{T}\right)$, for $i = 1, \ldots, p$.

3.2.4. Weighting the belief values generated by the single classifiers

From Step 2 of the proposed methodology, the outputs of the single classifiers $\mathbf{Y}_j = \{\mathbf{y}_{j1}, \ldots, \mathbf{y}_{jp}\}$ are available for each feature vector x_j of the test sample. As mentioned, we assume that $\mathbf{y}_{ji}$ is a vector of levels of belief in the assignment of x_j to each of the classes. Working with belief levels enables us to analyze the performance of a range of efficient classifier combination rules [69] in Step 4 of the proposed methodology. Step 3 consists in correcting these belief values by introducing the distributional distances d_i. Two different criteria are proposed.

One approach consists in assigning weights in inverse proportion to the energy distance for the corresponding node, i.e. $\omega_{ji} = K \cdot d_i^{-1}$ for all j, where $K = \left(\sum_{i=1}^{p} d_i^{-1}\right)^{-1}$, is a normalizing constant used to make the sum of weights equal to one. This way, the belief values generated from each local classifier receive a common weight for all instances in the test set, resulting $\mathbf{y}_j^{\omega} = K\, d_i^{-1}\mathbf{y}_j$, for all $j = 1, \ldots, t$. Hereafter this weighting approach will be referred to **per-Node Weighting** and denoted by **pNW**.

In order to provide a finer grain approach where the belief degrees per instance in the test set receive different weights, an alternative weighting approach is proposed. The aim is to assign weights in order to minimize the energy distance between each training sample and the test set. The procedure can be understood as if, for each node $\mathcal{N}_i$, a weighted resampling scheme of the test set is carried out to overweight belief values associated to instances better represented at the node than in the test sample. Features x_j allocated in low probability zones in the test set but belonging to high probability zones in a specific node will receive high weights, and conversely instances with low probability in the test set but high probability in the node will be downweighted (see Figure 3.2). By assigning high weights to instances with low representation in the test set but well-represented at the node, we ensure an efficient use of the training samples. Notice that equation (3.2) in Section 3.2.1 leads to theoretical weights $\omega_{ji} = P\left(x_j \,|\mathcal{N}_i\right) / P\left(x_j\right)$, thus accounting for the rationale of this approach. Unlike the per-node belief approach, under this new weighting criterion each node produces a weight for each instance in the test set. For this reason, this weighting approach will be referred as **per-Instance Weighting** and denoted by **pIW**.

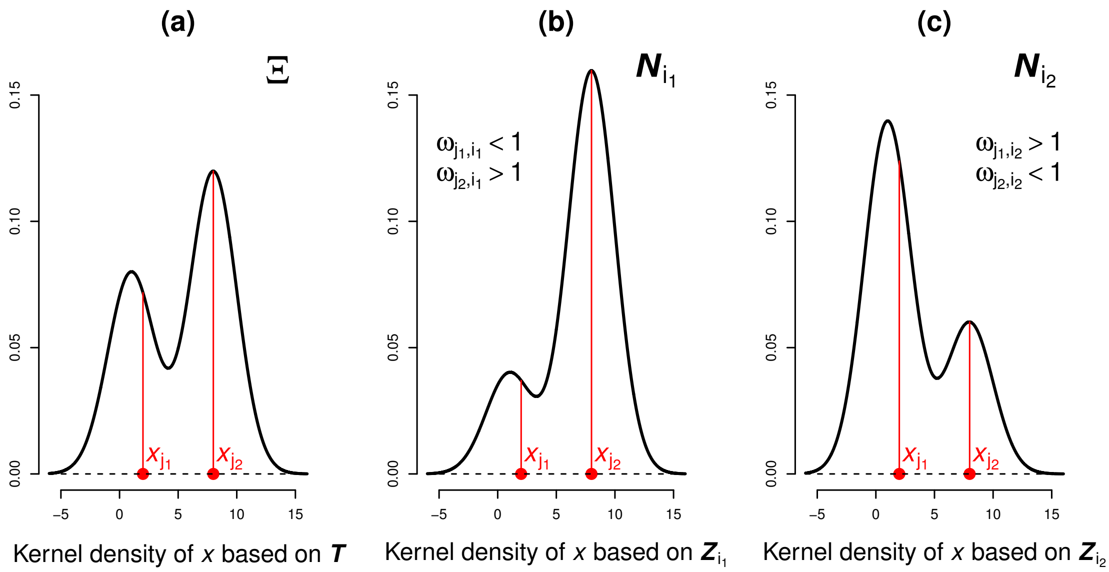


Figure 3.2: Graphical illustration of the per-Instance Weighting (pIW) criterion.

According to the definition of the energy distance in (3.3), the per-instance weights for the set of test instances at the i-th node, $\omega_i = (\omega_{1i}, \ldots, \omega_{ti})$, are obtained by minimizing the objective function $E(\omega_i)$ given by

$$E(\omega_i) = 2D_{\mathcal{Z}_i,\mathcal{T}}\omega_i^T - D_{\mathcal{Z}_i,\mathcal{Z}_i} - \omega_i D_{\mathcal{T},\mathcal{T}}\omega_i^T, \tag{3.4}$$

where $D_{\mathcal{A},\mathcal{B}}$ is the matrix whose (u,v)-element is $D_{\mathcal{A},\mathcal{B}}(u,v) = \|a_u - b_v\|$, for arbitrary datasets $\mathcal{A} \equiv (a_1, \ldots, a_r)$ and $\mathcal{B} \equiv (b_1, \ldots, b_s)$.

In practice, the minimization of $E(\omega_i)$ is posed by means of the optimization problem:

$$\begin{aligned} \underset{\omega_i}{\text{minimize}} \quad & \frac{1}{t} D_{\mathcal{N}_i,\mathcal{T}}\omega_i^T - \omega_i D_{\mathcal{T},\mathcal{T}}\omega_i^T \\ \text{subject to} \quad & \sum_{i=1}^{p} \omega_i = 1,\ \omega_i \succeq 0. \end{aligned}$$

3.2.5. Combining the belief values generated by the single classifiers

Last step in the proposed methodology consists in combining the weighted outputs of the single classifiers trained at each of the nodes, namely the vectors of belief degrees $\mathbf{y}_{ji}^{\omega} = \omega_{ji}\mathbf{y}_{ji} = (\omega_{ji}y_{ji1}, \ldots, \omega_{ji}y_{jim})$, whose k-th element $y_{jik}^{\omega} = \omega_{ji}y_{jik}$ provides an estimate of the posteriori probability $P\left(C_k \left| x_j, \mathcal{N}_i \right.\right)$, for $k = 1, \ldots, m$. Having available continuous outputs in form of belief values allows us to consider different functions of these values, so-called *decision rules* [98], to get a unique output. Kittler et al. [69] argue that the decision rules provide a useful approach to circumvent the complex problem of inferring the posteriori probability function

$$P\left(x_j \text{ is assigned to the class } C_k \left| \mathbf{y}_{j1}, \ldots, \mathbf{y}_{jp} \right.\right),$$

which would allow us to determine the most likely class using the Bayesian theory. Some alternative classifier combination approaches include techniques such as Stacked Generalization, Meta-Learning, Knowledge Probing and Effective Stacking [98]. Nevertheless, these methods work training a new classifier based on single outputs produced by each node, which requires access to a common training set or sharing of private training information among nodes, thus limiting their applicability and violating the condition of no communication between nodes. Supported by these arguments, we propose to use some of the most popular decision rules to generate the final assignment.

Following Kittler et al. [69], where a common theoretical framework for different decision rules is provided, we have considered in our experiments the set of rules presented below. In all cases, we assume that the belief values have been normalized so that $P\left(C_k \left| x_j, \mathcal{N}_i \right.\right) = y_{jik}^{\omega} / \sum_{l=1}^{m} y_{jil}^{\omega}$, for all j and i.

- *Product rule.* The instance with observed feature vector x_j is assigned to the class C_k if

$$\prod_{i=1}^{p} y_{jik}^{\omega} = \max_{1 \leq l \leq m} \prod_{i=1}^{p} y_{jil}^{\omega}.$$

  Note that, under this rule, a class with a zero or very small belief value from only one node will receive a zero or very small combined belief degree, even if the rest of nodes provide high belief degrees to the mentioned class. Hence, this rule will exhibit a bad performance if for example a class is not represented in a particular node.

- *Sum rule.* The instance with observed feature x_j is assigned to the class C_k if

$$\sum_{i=1}^{p} y^{\omega}_{jik} = \max_{1 \le l \le m} \sum_{i=1}^{p} y^{\omega}_{jil}.$$

The theoretical support for the sum rule lies on assuming that the posteriori probabilities do not deviate greatly from the prior probabilities [69], that is $P\left(C_k \left| x_j, \mathcal{N}_i \right.\right) = P\left(C_k\right) + \varepsilon_{jk}$, with ε_{jk} taking very small values for all k and j. Kittler et al. [69] have shown that the sum rule is less sensitive to the estimate errors than the product rule.

- *Max rule.* The instance with observed feature x_j is assigned to the class C_k if

$$\max_{1 \le i \le p} y^{\omega}_{jik} = \max_{1 \le l \le m} \max_{1 \le i \le p} y^{\omega}_{jil}.$$

The class obtaining the highest belief degree over all the nodes is selected as combined output. It can be shown that this rule approximates the sum rule under the assumption of equal prior probabilities for the classes.

- *Min rule.* The instance with observed feature x_j is assigned to the class C_k if

$$\min_{1 \le i \le p} y^{\omega}_{jik} = \max_{1 \le l \le m} \min_{i=1}^{p} y^{\omega}_{jil}.$$

Assuming as before that classes are a priori equiprobable, the min rule approximates the product rule.

- *Majority vote rule.* The instance with observed feature x_j is assigned to the class C_k if

$$\sum_{i=1}^{p} \Delta_{jik} = \max_{1 \le l \le m} \sum_{i=1}^{p} \Delta_{jil},$$

where $\Delta_{jil} = 1$ if $y^{\omega}_{jil} = \max_{1 \le u \le m} y^{\omega}_{jiu}$ and $\Delta_{jil} = 0$ otherwise. Therefore, the combined output consists in selecting the class receiving the largest number of votes from the single classifiers. Under the equiprobability assumption for the prior probabilities, this rule matches the sum rule when the belief values are discretized by using the Δ_{jil} values.

3.2.6. Some remarks

Some remarks concerning the proposed methodology are highlighted below.

Remark 1. The estimated distributional distance d_i between a particular node $\mathcal{N}_i$ and Ξ could be small (large) even though a few test instances are bad (well) represented at $\mathcal{N}_i$. In any case, all the classifier outputs obtained at $\mathcal{N}_i$ will receive the same weight when the pNW criterion is used. This is an unsuitable consequence of taking weights based on the global distance d_i such as pNW does. On the contrary, the pIW criterion checks the point-to-point distribution matching, thus being sensitive to local deviations. Note that if a feature vector x_j is badly represented at a specific node, then it must be well represented at another node because $\mathcal{P}$ is a partition of the feature domain. In sum, the pIW criterion is expected to outperform the pNW one, and the improvement would be more substantial with unbalanced nodes. In our experimental evaluation in Section 3.4.2, both weighting criteria are examined and compared with a standard approach without weighting the single belief values (an unweighted approach denoted by **UW**). A scheme of the three distributed approaches is shown in Figure 3.3.

Remark 2. In a non-distributed classification context with different distributions for the training and test sets (*sample selection bias* problem), Huang et al. [61] proposed to use the unlabeled data to reweight the training data in such a way that the means of the training and test features in a reproducing kernel Hilbert space are close. Although in a different context, this is a similar idea to the pIW approach and it is worthy to emphasize the main differences. In our work, the reweighting process is applied to the test data because the nodes cannot be retrained in our distributed scenario. On the other hand, a key assumption in [61] is that the conditional probability of $C|x$ is the same for the training and test populations so that the bias is only exhibited by the feature distributions. In our framework, stationarity is assumed and therefore the bias can only be present between nodes. Nevertheless, it is not necessary to require that the conditional probabilities of $C|x$

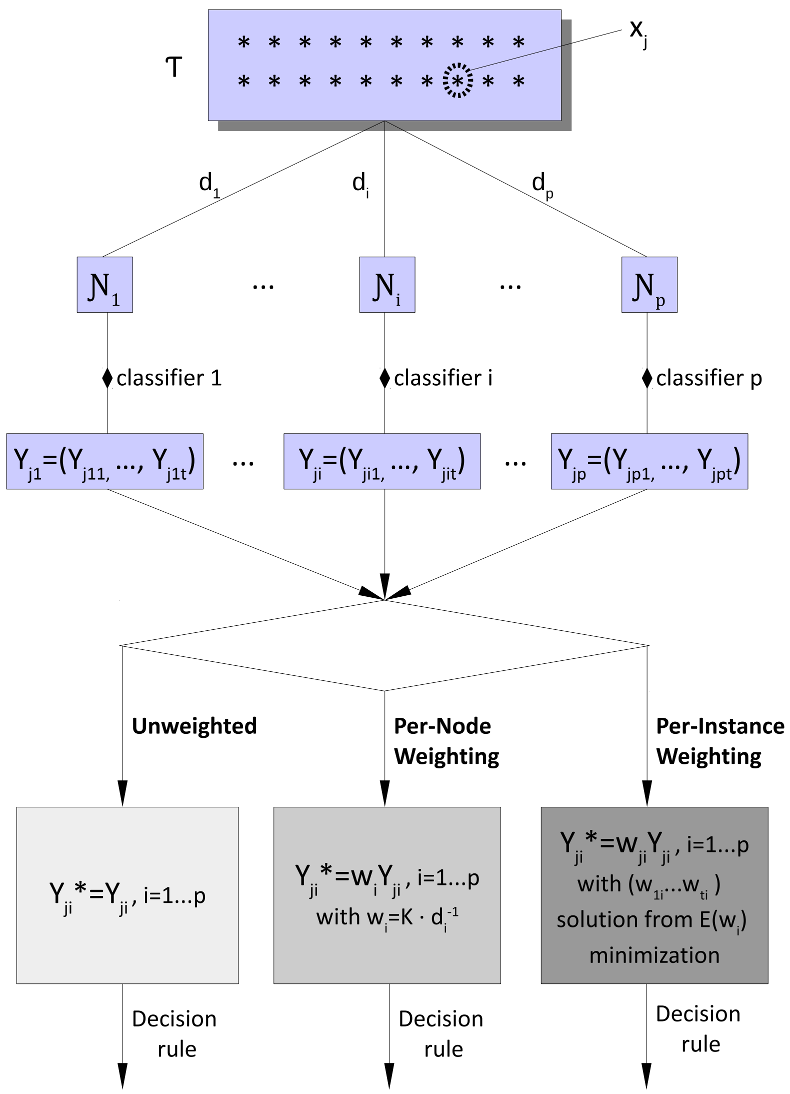


Figure 3.3: Distributed approaches schemes.

remain unchanged across the nodes, which would be a very restrictive constraint.

Remark 3. As already mentioned, Kittler et al.[69] pointed out some nice properties of the sum rule to combine the single classifier outputs. Beyond these properties, equation (3.2) provides theoretical support to use this criterion since the posteriori probabilities are expressed as a weighted sum of the single classifier outputs within each node.

Remark 4. The proposed learning model is not restricted to the use of a particular classification model at each node. The unique requirement is that the classifier outcome consists of a vector of belief values or posteriori probabilities of the classes for a given feature vector. Thus, artificial neural network, logistic regression, support vector machines, Bayesian classifiers, and Random Forest could be used among others.

3.3. Experimental settings

The main characteristics of the experiments are detailed below.

Classifiers. To study the interaction between the distributed learning models and the classifier type, five classification algorithms are considered, namely Random Forest (RF), a support vector machine with RBF Kernel (SVM), the Fisher's linear discriminant (LDA), the classifier based on multinomial logistic regression (Mult), and the XGBoost (eXtreme Gradient Boosting) algorithm (XGB), a fast implementation of the gradient boosting using decision trees. All of them were executed by using different R packages, `randomForest` [74] for RF, `e1071` [84] for SVM, `xgboost`[29] for XGB, and `MASS` and `nnet` [124] for LDA and Mult, respectively. The default parameters are taken in all cases since our concern is not to determine the most efficient inputs but comparing the models under homogeneous conditions. All classifiers provide the options to output belief values in addition to classes.

Datasets. Seven datasets are used to analyze the coupling between the proposed method and the underlying classification problem. Five databases (Spambase, KDD Cup 99, Connect-4, Covertype, and Higgs) contain real data and are available from the UCI Machine Learning Repository [75]. The other two databases (Simul-C2 and Simul-C8) consist of synthetic data generated from simulated classification scenarios. The main characteristics of these datasets are summarized in Table 3.1, including the total number of instances, the dimension d of the feature space, and the number m of classes forming $\mathcal{C}$.

To get a quick understanding on the nature of these datasets, a very brief description of each one is provided below.

- KDD Cup 99. Benchmark dataset in the intrusion detection field, which

Table 3.1: Datasets characteristics.

Dataset	#Samples	#Features	#Classes
Spambase	4601	57	2
KDD Cup 99	825,050	41	5
Connect-4	67,557	42	3
Covertype	581,012	54	7
Higgs	100,000	28	2
Simul-C2		5	2
Simul-C8		3	8

contains 5 million instances featured by 41 attributes and 39 types of distinct attacks, grouped into four classes of attack (DoS, Probe, R2L and U2R) and one class of non-attack (normal pattern) [1]. In our study, a smaller subset with 494,021 instances is used as training sample (10% of the original training set). For the test set, we used a subset of 331,029 patterns including new attacks that are not present in the training set. Around 20% of the two datasets are normal patterns (no attacks). The percentages of class labels for the training and test sets are shown in Table 3.2. As can be seen, the percentage of attacks in both datasets is very high, overcoming 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) formed by very few instances. Due to these characteristics, KDD Cup 99 becomes a real challenge for the classification task.

Table 3.2: Distribution (in percentage) of normal activities and kinds of attacks in KDD Cup 99 dataset.

Type	Training 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

- Simul-C2. Consider a square grid of size 3 in dimension 5 and, centered at each grid node, a 5-dimensional Gaussian distribution with uncorrelated components of equal variance 0.05^2. Each Gaussian is assigned to one of $m = 2$ possible classes at random. In this scenario, an identical number of data are drawn out from each Gaussian to form our first synthetic dataset. Figure 3.4 provides an intuition on the structure of Simul-C2 in dimension 2. This scenario lets us have an exact knowledge of the complexity of the classification task in order to derive some insight into the results.

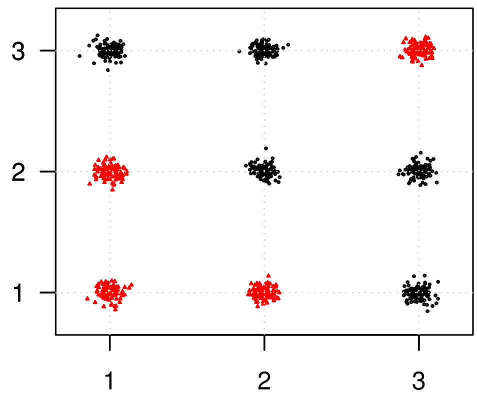


Figure 3.4: Plot of a simulated trial from a 2-dimensional version of Simul-C2 scenario. Color identifies the class.

- Simul-C8. Synthetic dataset generated in a similar way as Simul-C2, but now with 3-dimensional Gaussian distributions randomly assigned to $m = 8$ classes.

Sample size. At each experimental trial, the sizes of both the training sample $\mathcal{Z}$ and the test sample $\mathcal{T}$ are fixed to 500, i.e. $n = t = 500$. The training sample is then equidistributed between the nodes so that $n_i = 500/p$, for all $i = 1, \ldots, p$.

Balancedness. Since no constraints on the fragmentation scheme are imposed, it is interesting to check the behavior of our learning model with balanced and unbalanced nodes, i.e. nodes exhibiting similar or different distributions, respectively.

The balanced scenarios are recreated by allocating instances to each node at random and without replacement while maintaining the class proportions. The unbalanced scenarios are set up as follows. First, one node with the same class proportions as the entire training set is formed. For the rest of nodes, the class proportions are perturbed by multiplying each one of them by a random number uniformly generated between 0.3 and 1.7, and then normalizing. In consecutive nodes, the overall class proportions are updated on the basis of the number of remaining training instances, and sampling without replacement is always carried out.

Partition size. To assess the classification accuracy as data fragmentation increases, the training set was randomly split into 2, 4, 7, 11 and 15 nodes. The unique randomization restrictions are imposed by the class proportions at each node, which depend on whether a balanced or unbalanced scenario is considered.

Decision Rules. The belief values generated by the classifiers at each node are combined according to the five decision rules enumerated in Section 3.2.5, namely the Product, Sum, Max, Min and Majority rules.

3.4. Experimental results

An empirical study addressed to motivate and evaluate the performance of the proposed learning models has been carried out. An overview and discussion of the main results are presented in this section.

3.4.1. Some motivating experiments

By construction, the proposed learning models take into account the distances between the probability distributions of the features in the population and within each node. The heuristic is that, in general, smaller distributional distances between training and test sets tend to produce better classification results. Indeed, the key issue is how these distances should be jointly used to attain this improvement. Beyond this issue, a pair of motivating experiments designed to provide empirical support for this heuristic have been carried out. The first experiment consisted in checking for the existence of negative correlation between distributional distance and classification accuracy. In the second one, a distributed scenario is considered, and then the proportion of times that the node with the smallest distributional distance produces the best classification accuracy is measured.

For these specific experiments, the Spambase dataset is used and the within-node distributions are generated according to the unbalancing approach described in Section 3.3. The number of nodes is set to $p = 5$ and a training sample of size $n_i = 200$ is used at each node to train a Random Forest classifier.

Considering test samples with the same size, $t = 200$, the first experiment consisted in measuring the distributional distances between training and test samples using the energy distance d_i introduced in Section 3.2.3, and simultaneously computing the proportion of test data correctly classified at each node. This process was performed for a large number of trials, and the outputs are plotted in Figure 3.5. A clear negative correlation between distributional distance and classification accuracy is observed, thus supporting the argument that less distant nodes tend to produce better classification accuracy.

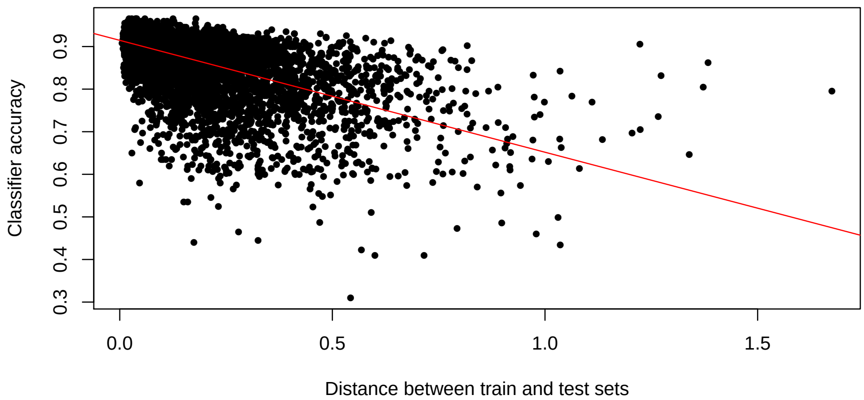


Figure 3.5: Correlation between distributional distance and classifier accuracy.

In the second experiment, the test sample size is not constant at all trials, taking values moving from 200 to 3200. Notice that the distributional distance becomes more accurately approximated as test sample size increases, and therefore the classification accuracy should be also higher. In a five node distributed scenario, the expected proportion of times that a node picked at random produces the best classifier is 0.2. Table 3.3 shows the proportion of times that the node with the smallest distributional distance produced the highest classification accuracy in our experiment, denoted by $p_{\min(d_i)}$. It is observed that the node with the smallest distributional distance becomes the best one in an increasing proportion with the test sample size, always above the baseline proportion 0.2, until it is approximately doubled.

Table 3.3: Proportion of times that the smallest distributional distance leads to the best classifier node ($p_{\min(d_i)}$) against the test sample size (t).

t	200	1200	2200	3200
$p_{\min(d_i)}$	0.28	0.35	0.37	0.37

In sum, these first experiments empirically illustrate the interest in distributed learning models regarding distributional distances between training nodes and test samples. We propose models taking into consideration this principle, but in addition, they take advantage of combining efficiently the classifiers produced by each node instead of simply selecting one of them.

3.4.2. Results

The accuracy of the two weighting criteria (pNW and pIW) described in Section 3.2.4 was checked on all the combinations of parameters involved in our experimental setup, namely classifiers, datasets, partition sizes, decision rules, and balanced and unbalanced scenarios (Section 3.3). For comparison purposes, accuracy results based on a standard unweighted distributed model (UW) and a non-distributed model (ND) were also obtained. The ND model uses the entire dataset $\mathcal{Z}$ to train a unique classifier. Hence, ND is expected to achieve the highest classification accuracy, and its results can be taken as an upper reference level. Besides the classification accuracy, values of precision and recall were also evaluated.

For each combination of parameters, the experiment was replicated $N = 300$ times and average classification accuracy values were obtained for each learning model. In order to examine how the learning models interact with the different parameters, the average results were aggregated in different ways. For example, Table 3.4 shows the average accuracy attained with each classifier. It is observed that the weighted models interact better with SVM, LDA and Mult than with Random Forest and XGBoost in the unbalanced setting (see Figure 3.6). In particular, the most significant improvement rates due to the pIW model in the unbalanced setup are observed for Mult and SVM. In the latter case, this may be connected with the fact that SVM with Gaussian kernel and energy distance are based on Euclidean inter-point distances.

Table 3.4: Average classification accuracy values conditional on classifier type. Rows under the last sub-table (MEAN) show the averages over all trials, including balanced and unbalanced scenarios. The lack of results for ND in the first two sub-tables is due to the ND model assumes non-distributed data, i.e. no partitions (balanced or unbalanced) are considered.

	Classifier				
Model	RF	SVM	XGB	LDA	Mult
BALANCED					
pNW	0.7087	0.5852	0.6923	0.5768	0.6084
pIW	0.7173	0.5908	0.7016	0.5826	0.6168
UW	0.7131	0.5881	0.6964	0.5769	0.6066
UNBALANCED					
pNW	0.6935	0.5804	0.6843	0.5727	0.6035
pIW	0.7059	0.5921	0.6977	0.5863	0.6186
UW	0.6996	0.5784	0.6909	0.5749	0.6022
MEAN					
pNW	0.7011	0.5828	0.6883	0.5747	0.6060
pIW	0.7116	0.5915	0.6997	0.5845	0.6177
UW	0.7063	0.5832	0.6937	0.5759	0.6044
ND	0.7566	0.6719	0.7398	0.6315	0.6423

The average results aggregated by decision rule are reported in Table 3.5 and graphically represented using bar charts in Figure 3.7. Regardless of whether the partitioning is balanced or not, the SUM rule produces the best average results with the three distributed models. This result is consistent with the experimental findings in Kittler *et al.* [69] and with our theoretical arguments in Section 3.2 (Remark 3

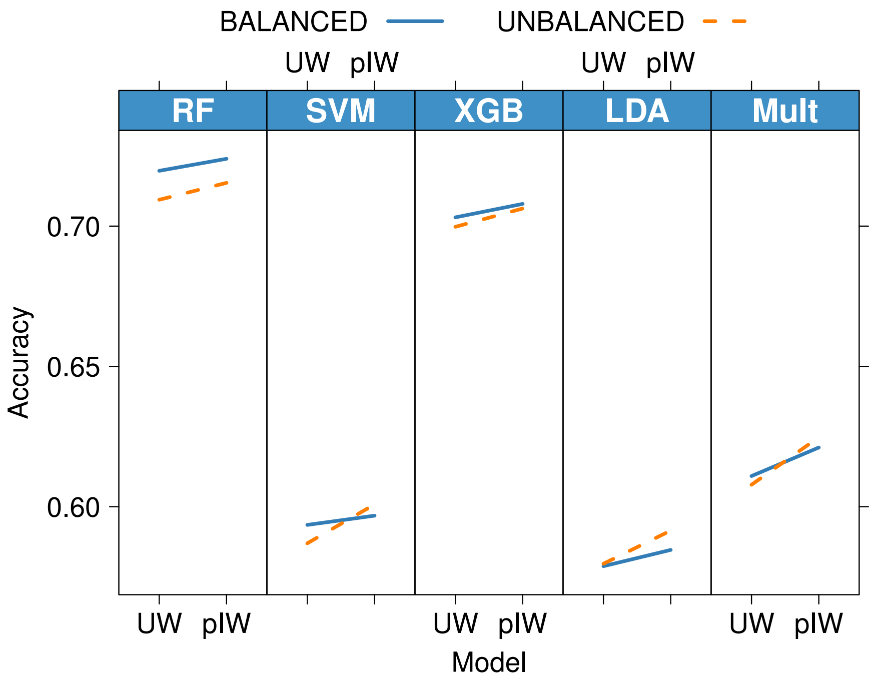


Figure 3.6: Accuracy-based interaction plot to check the joint effect of classifier, learning model and scenario.

in Section 3.2.6).

Table 3.5 and Figure 3.7 also allow to compare the average accuracy attained with the different models. Except for the MIN and PROD rules, the highest accuracy values are obtained with the per-Instance Weighting. Nevertheless, the MIN rule fairly produces the worst results and no differences between weighting approaches are observed with the PROD rule. Therefore, it is concluded that the per-Instance Weighting approach fairly leads to the best results.

Overall, pNW performs worse than UW on average. This behavior is somewhat surprising in the light of the results showed in the motivating experiments of Section 3.4.1. We guess that this may be caused by the joint effect of two circumstances, namely the global character of the per-Node weights (see Remark 1 in Section 3.2.6) and the noise increase generated by the variability of these weights (Figure 3.5 illustrates this variability).

The average accuracies aggregated by partition size are shown in Figure 3.8.

Table 3.5: Average classification accuracy values conditional on the decision rules.

	Decision rule					
Model	MAJ	MAX	MIN	PROD	SUM	Mean
BALANCED						
pNW	0.6442	0.6252	0.6184	0.6347	0.6491	0.6343
pIW	0.6582	0.6461	0.6066	0.6345	0.6639	0.6418
UW	0.6456	0.6283	0.6205	0.6347	0.6519	0.6362
Mean	0.6493	0.6332	0.6152	0.6346	0.6549	0.6374
UNBALANCED						
pNW	0.6407	0.6191	0.5977	0.6276	0.6493	0.6269
pIW	0.6652	0.6513	0.5823	0.6275	0.6743	0.6401
UW	0.6454	0.6189	0.6008	0.6277	0.6531	0.6292
Mean	0.6504	0.6297	0.5936	0.6276	0.6589	0.6321
ND	—	—	—	—	—	0.6884

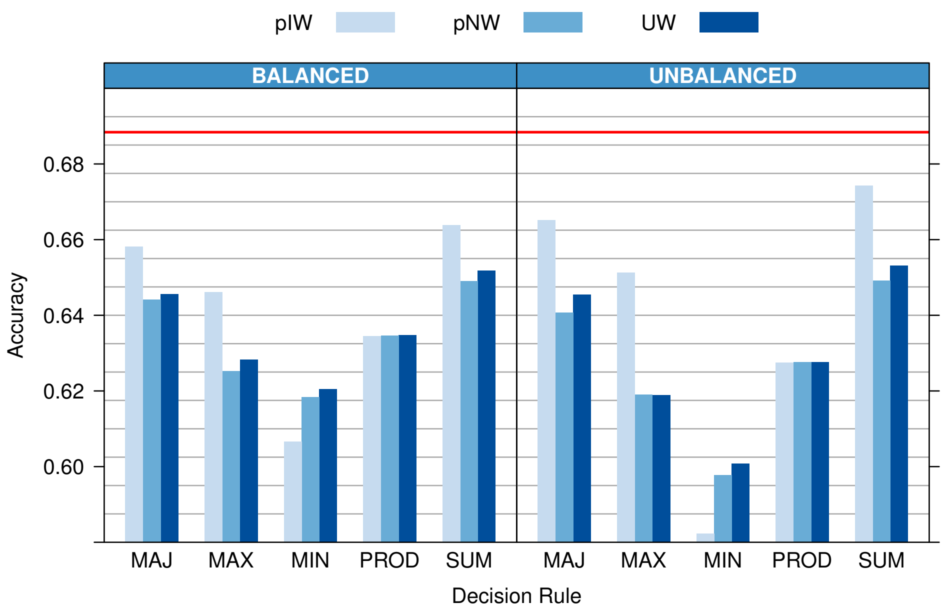


Figure 3.7: Average classification accuracy values aggregated by decision rules. The horizontal red line indicates the average accuracy for the ND model.

Significant degradation of accuracy with the number of nodes is evident for all combinations of decision rule and learning model in balanced and unbalanced scenarios.

For all the models, the MIN rule degrades faster with fragmentation, although it is very competitive with UW and pNW for a small number of nodes. As the best-performing pIW approach is considered, the MIN rule is substantial and uniformly the worst decision rule. SUM, MAJ and MAX exhibit similar performance, with SUM having a slight edge. The good behavior of pIW deserves particular attention. Note that, except for the MIN rule, the pIW approach always produces the highest percentages of correct classification for all the levels of fragmentation regardless of the used rule.

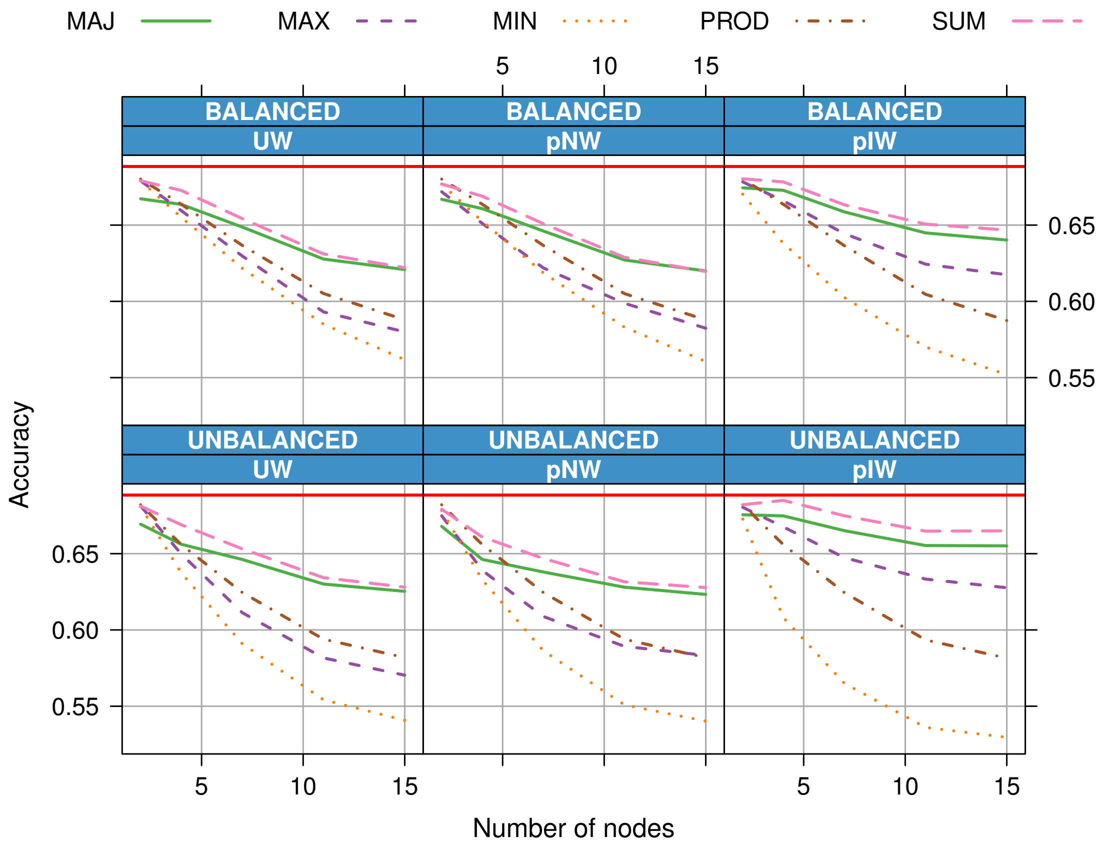


Figure 3.8: Average accuracy as function of the partition size. The horizontal red lines indicate the average accuracy for the ND model.

All our results allow to conclude that the favorable effects of the weighting approaches are more important in unbalanced scenarios. In particular, the amount of accuracy improvement produced by the per-Instance Weighting model is clearly stronger when unbalancing.

Figure 3.9 shows separately the average results for each dataset and provides additional insight into the behavior of the proposed models. Concerning the KDD

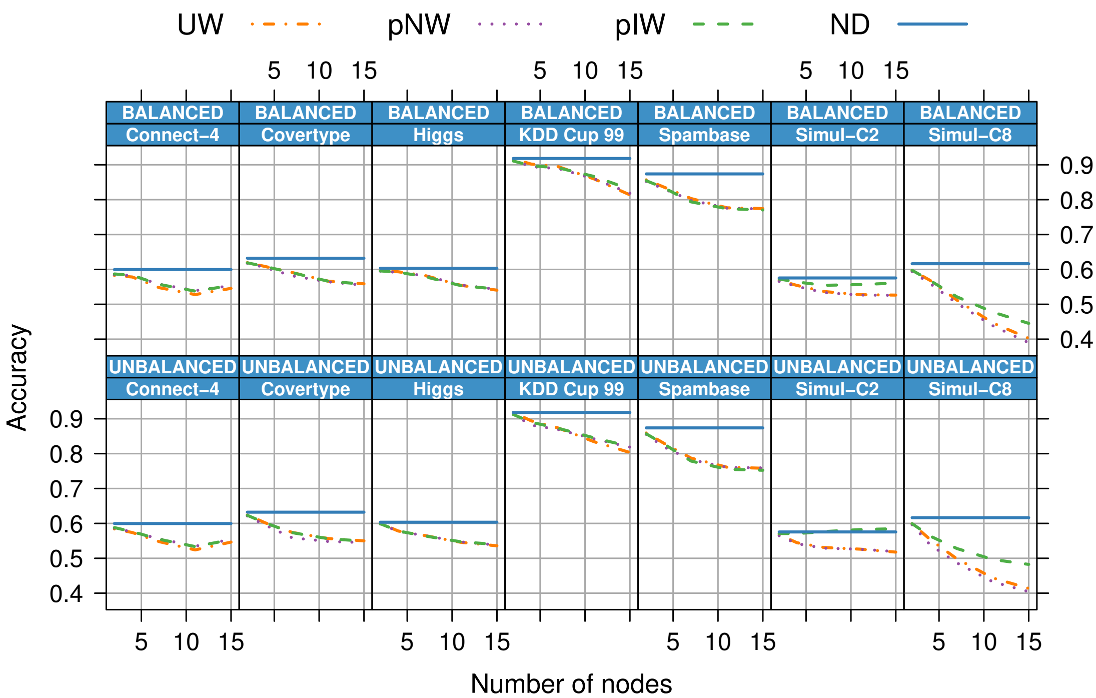


Figure 3.9: Average accuracy as function of the partition size for each dataset. The horizontal blue lines indicate the average accuracy for the ND model.

Cup 99 dataset, it is noticeable the good performance showed by the per-Node Weighting approach in unbalanced scenarios. In fact, pNW and pIW behave very similarly and fairly outperform the Unweighted model. Since KKD Cup 99 exhibits concept drift, this result illustrates that our distributed approaches work well for various types of differences in distribution between nodes and population, no matter how these differences occur. In other words, the effectiveness of our proposal is not restricted to the case of distributional differences caused by a non-uniform partition of the data.

The pIW model reports lower accuracy with Spambase, while it performs slightly better than the other distributed models in Connect-4, Covertype and Higgs, the most complex scenarios in terms of classification. In these cases, the non-distributed model only reaches an accuracy around 0.6, and the distributed approaches are reasonably close to this proportion for all partition sizes. An atypical behavior is observed for Connect-4 since the classification accuracy does not monotonically decrease with the number of partitions. So, in this particular case, it looks more likely that the local scenarios generated by splitting the data lead to an easier classification task.

In addition to knowing the exact underlying distributions, the analysis of simulated data is free of limitations to generate nodes maintaining the required regularity and provides insight into the level of difficulty. The results for our simulated datasets, Simul-C2 and Simul-C8, are particularly interesting. In both cases, pIW clearly draws the best results, and the differences are more substantial in the unbalanced setting. In the scenario with only two classes, degradation with the number of nodes is almost prevented in the balanced setting, and the results end up surpassing the non-distributed approach in Simul-C2. In the non-distributed approach, a linear classifier will have bad performance in this scenario, since the classification frontier is non linear. When distributed, different local models work better. In Simul-C8, degradation of pIW with the fragmentation is more marked, but still less severe than in the case of pNW and UW.

3.5. Computational Cost

In addition to the cost of training each classifier, our approach requires calculating the distributional distance between each node and the test set, and the pIW model requires each node to calculate the weights that minimize this distance.

The cost of training the classifiers depends on the chosen method, but the improvement is equivalent to reducing the training sample by the number of nodes, e.g. if a given classifier trains in $\mathcal{O}(n^3)$ then the complexity will be reduced to $\mathcal{O}(n^3/p^3)$. Similarly, calculating the energy statistic has computational complexity $\mathcal{O}((n/p)^2 + t^2 + (n/p)t)$, given n the training size, p the number of nodes that the training set is fragmented into and t the test size. Even though it can be considered a constant factor, we introduce the number of nodes p to highlight the strong computational benefits of this distributed approach.

Finding the individual weights for the pIW approach has the complexity of solving the related quadratic programming problem, and is the usually dominating complexity: Experimentally, this quadratic programming complexity is between $\mathcal{O}(t^2)$ and $\mathcal{O}(t^3)$. The influence of the training test size on the quadratic programming complexity is linear: $\mathcal{O}(n/p)$.

Assuming the underlying classifiers test in constant time, the complexity of classifying a given test set goes from $\mathcal{O}(t)$ to a worst case of $\mathcal{O}(t^3 + n)$ for the pIW version.

3.6. Summary

In a distributed classification framework, we have proposed two weighted approaches that combine local classifiers trained at each node to improve overall classification accuracy. The two approaches assume the availability of a test set and are based on the distance between the distributions of the feature vectors of each node and the test set. The first approach, per-Node Weighting, assigns the same weight at each node to all test instances, while the per-Instance Weighting approach achieves finer granularity by allowing distinct weights for each test instance at each node.

Under the general assumption that both the test set and the entire training set are i.i.d. samples from the population in study, we have motivated the proposed weighting criteria and provided theoretical support for the optimality of combining the classifier outcomes using a weighted sum. Our framework makes no assumptions about the structure or distribution of the data across the nodes. In fact, by construction, our classification models are particularly useful to deal with heterogeneity of data among the nodes, which usually happens in real-world distributed datasets. In addition, our technique requires no communication between nodes, preserving data privacy, allowing combination of different classifier models and maximizing computational efficiency.

Our experimental study involving synthetic and real datasets has illustrated the good performance of the proposed models compared to standard classifier combination rules. Overall, the per-Instance Weighting approach achieves the best results. As expected, the improvement is more substantial when treating with unbalanced nodes, under all tested classifiers an partition sizes. Our experiments also illustrate that the sum rule outperforms other alternative decision rules. The per-Node Weighting approach does not achieve improvement over the standard approaches but on the most extreme cases when the individual nodes training sets differ the most.

Chapter 4

Distributed feature selection

To confront the problem of the extremely high dimensionality it is advisable to investigate the effects of the application of feature selection. The use of an adequate feature selection method can avoid over-fitting and improve model performance, providing faster and more cost-effective models and a deeper insight into the underlying processes that generated the data [106]. However, as stated in previous chapter for learning methods, we will have to deal with a scalibility problem if we apply these techniques to large datasets due to their high computational complexities. Therefore, a possible solution to reduce the complexity of feature selection process in the current big data scenario might be to distribute the data, run a feature selection method on each subset of data and then combine the results. As mentioned in the introduction of Chapter 3, the partition can be done vertically or horizontally. Both alternatives will be explored in this chapter since horizontal partitioning is especially suitable when dealing with datasets with a high number of samples whilst vertical partitioning is more appropriate to datasets with a high number of features.

In this chapter, we will present a methodology in which several rounds of feature selection are performed on different partitions of the data. Then, the partial outputs are combined into a single subset of relevant features according to the theoretical complexity of these features using several data complexity measures. We (a) include experimental results for both horizontal and vertical partitioning strategies; (b) perform a more intensive study, making use of several complexity measures to combine the partial rankings of features; (c) include datasets of different sample sizes, with different number of classes, and also microarray datasets, trying to check the behavior of the approach on high input dimensionality and (d) examine the effects of including different levels of overlap in the feature subsets.

4.1. Background

Although distributed learning is a fairly new field, it has been receiving a growing amount of attention since its inception. There exist in the literature several works to scale up datasets that are too large for machine learning in terms of features. In these works [83, 111], authors described a novel ensemble approach, in which data is partitioned by features. Results show that this technique is simple, predicts almost as well as a centralized approach, reduces the amount of communication required, distributes computation and data access well, and allows each local site to keep its raw data private. Kamalika et al. [31] developed a local distributed privacy preserving algorithm for feature selection in large peer-to-peer environment. Banerhee et al. [9] proposed a distributed privacy preserving method to perform feature selection that handles both horizontal and vertical data partitioning, whose efficiency has been demonstrated for real life datasets including time series data. Besides, several paradigms for performing distributed learning have emerged in the last decade. Peralta et al. [97] present a feature selection method based on evolutionary computation which uses the MapReduce paradigm to obtain subsets of features from big datasets. Developed in Spark, Eiras-Franco et al. [43] proposed distributed versions of four popular feature selection algorithms. Ramírez et al. [102] also use the Apache Spark paradigm to implement a distributed version of generic feature selection framework that includes a broad group of well-known information theory-based methods.

In Bolón-Canedo et al. [21], we presented a methodology for distributing the data vertically which combined partial feature subsets based on improvements in classification accuracy. Although the experiments showed that execution time was considerably shortened whereas performance was maintained or even improved compared to standard algorithms applied to the non-partitioned datasets, the drawback of this methodology was its dependence on the classifier used. In order to overcome this issue, a new framework for distributing the feature selection process [18, 86] was proposed, which performed a merging procedure to update the final feature subset according to the theoretical complexity of these features, by using data complexity measures instead of the classification error. Such measures provides a basis for analyzing classifier performance beyond estimates of error rates (see Section 2.2 in Chapter 2 for more details). In this way, we provided a framework for distributed feature selection which not only was independent of the classifier, but also reduced drastically the computational time needed by the algorithm, thus paving the way for its application in high dimensional datasets.

4.2. Distributed feature selection based on complexity measures (DFS-CM)

Our proposed framework for distributed feature selection (DFS-CM) can be summarized in the three following stages:

1. Partition of the training datasets in several packets (by samples or features).
2. Application of the distributed algorithm to the subsets in several rounds.
3. Combination of the results into a single feature subset.

The pseudocode for the distributed algorithms is shown in Algorithm 1 (horizontal partitioning) and Algorithm 2 (vertical partitioning). For each of the iterations of our methodology, which we call "rounds", the first step is to randomly partition (by samples or features) the training dataset D into a number of disjoint packets. Repeating the process in several rounds (see lines 2–10 in Algorithms 1 and 2) ensures that we have gathered enough information for the final combination step. Then, the feature selection method is applied with no adjustments to each of these partitions separately—which could be done in parallel, as all of them are independent to each other—and the features selected to be removed receive a vote. Note that these algorithms can be used with any feature selection method, although the use of filters is highly recommended since they are faster than other techniques. Note that the proposed method DFS-CM can be also applied on ranker methods, another threshold has to be established, however, to determine the number of features to be removed in each subset of data.

At this point, a new round is performed leading to a new partition of the dataset, followed by another iteration of votes. Finally, after the predefined number of rounds, the features that have received a number of votes above a certain threshold are removed, and the remaining feature subset will be used in the training and test sets. In order to determine the optimal threshold of votes, we followed the recommendations exposed in Haro-García et al. [58] for our first approach [21], where the best value will be the one that minimizes the training classification error and the percentage of features retained to the extent possible:

$$e[v] \leftarrow \alpha \times error + (1-\alpha) \times featPercentage,$$

where α is a parameter with a value in the interval [0,1] which measures the relative relevance of both values. Different values can be used if the researcher is more interested in reducing features or in error.

The drawback of applying this approach is that, by involving a classifier in the process of selecting the optimal threshold, our methodology is dependent on the classifier chosen. Moreover, in some cases the required time for this task is higher than the time necessary for the feature selection process. Trying to overcome these issues, we proposed to modify the function for calculating the threshold of votes by making use of data complexity measures [18]. The reason for this decision was that we assume that good candidate features would contribute to decrease the theoretical complexity of the data and must be maintained. Since our intention was to propose a framework that could be independent of the classifier and applicable to both binary and multiclass datasets, the Fisher's multiple discriminant ratio for C classes was chosen (Eq. 2.2).

We used the inverse of the Fisher ratio $1/f$—from now noted as *F1*—where a small complexity value represents an easier problem. Therefore, the new formula for calculating $e[v]$ was defined as (see line 17 in Algorithms 1 and 2):

$$e[v] \leftarrow \alpha \times F1 + (1-\alpha) \times featPercentage$$

Apart from the way of partitioning the data, the interval $[minVote, maxVote]$ changes for the two techniques. In the horizontal approach (see lines 13–14 in

Algorithm 1 Pseudo-code for horizontal partitioning

Data: $D_{(m \times n+1)} \leftarrow$ labeled training dataset with m samples and n input features

X $\leftarrow$ set of features, X=$\{x_1, ..., x_n\}$

$s \leftarrow$ number of submatrices of D with p samples

$V \leftarrow$ vector of votes

$r \leftarrow$ number of rounds ▷ 5 in this experimentation

$\alpha \leftarrow$ relative relevance of *complexitymeasure* and *featPercentage*

Result: S $\leftarrow$ subset of features \S $\subset$ X

//* Obtaining a vector of votes for discarding features *//

initialize the vector of votes V to 0, $|V|$=n

for each round **do**

  split D into s disjoint maintaining the class distribution

  for each submatrix **do**

    apply a feature selection algorithm

    F $\leftarrow$ features selected by the algorithm

    E $\leftarrow$ features removed by the algorithm \E $\cup$ F = X

    increment one vote in vector V for each feature in E

  end for

end for

//* Obtaining a threshold of votes, *Th*, to remove a feature *//

avg $\leftarrow$ compute the average of the vector votes

std $\leftarrow$ compute the standard deviation of the vector votes

minVote $\leftarrow$ minimum threshold considered ▷ $(avg - 1/2std)$

maxVote $\leftarrow$ maximum threshold considered ▷ $(avg + 1/2std)$

for $v \leftarrow$ *minVote to maxVote with increment 5* **do**

  $F_{th} \leftarrow$ subset of selected features ▷ number of votes $< v$

  complexityMeasure $\leftarrow$ value of the data complexity measure computed on training dataset

  featPercentage $\leftarrow$ percentage of features retained ▷ $\frac{|F_{th}|}{|X|} \times 100$

  e[v] $\leftarrow \alpha \times$ *complexityMeasure* + (1-α) $\times$ *featPercentage*

end for

Th $\leftarrow$ *min(e)*, *Th* is the value which minimizes the function e

S $\leftarrow$ subset of features after removing from X all features with a number of votes $\geq Th$

Algorithm 1), trying to avoid a high number of calculations which could lead to unaffordable computing times—the maximum number of votes v_{max} in some cases

Algorithm 2 Pseudo-code for vertical partitioning

Data: $D_{(m \times n+1)} \leftarrow$ labeled training dataset with m samples and n input features

 X $\leftarrow$ set of features, X=$\{x_1, ..., x_n\}$

 $s \leftarrow$ number of submatrices of D with p samples

 $V \leftarrow$ vector of votes

 $r \leftarrow$ number of rounds ▷ 5 in this experimentation

 $\alpha \leftarrow$ relative relevance of *complexitymeasure* and *featPercentage*

Result: S $\leftarrow$ subset of features \S $\subset$ X

//* Obtaining a vector of votes for discarding features *//

initialize the vector of votes V to 0, $|V|$=n
for each round **do**
  split D into s disjoint submatrices with m samples and t features
  for each submatrix **do**
    apply a feature selection algorithm
    F $\leftarrow$ features selected by the algorithm
    E $\leftarrow$ features removed by the algorithm \E $\cup$ F = X
    increment one vote in vector V for each feature in E
  end for
end for

//* Obtaining a threshold of votes, *Th*, to remove a feature *//

minVote $\leftarrow$ minimum threshold considered ▷ 1
maxVote $\leftarrow$ maximum threshold considered ▷ number of rounds, r
for $v \leftarrow$ *minVote to maxVote with increment 1* **do**
  $F_{th} \leftarrow$ subset of selected features ▷ number of votes $< v$
  complexityMeasure $\leftarrow$ value of the data complexity measure computed on training dataset
  featPercentage $\leftarrow$ percentage of features retained ▷ $\frac{|F_{th}|}{|X|} \times 100$
  e[v] $\leftarrow \alpha \times$ *complexityMeasure* + (1-α) $\times$ *featPercentage*
end for
Th $\leftarrow$ *min(e)*, *Th* is the value which minimizes the function e
S $\leftarrow$ subset of features after removing from X all features with a number of votes $\geq Th$

might be in the order of thousands—we opted for delimiting an interval computed using the mean and standard deviation such that $minVote = avg - 1/2std$ and $maxVote = avg + 1/2std$ instead of evaluating all the possible values for the number of votes [18]. If in the future we need to deal with datasets with millions of data, it is better to reduce the computation as much as possible.

As expected, the experimental results obtained showed that the time required by the distributed methods was drastically reduced for all datasets (the first six included in Table 4.1) and feature selection methods if compared with that of the centralized approach [18]. Since our first distributed approach D-Clas made use of the classification error to establish the threshold, the time required depended highly on the classifier, whilst the second distributed approach based on a complexity measure (D-Comp) this time was independent of the classifier. Therefore, in Figure 4.1, we can see the values of speed up —which indicate the performance improvement of D-Comp with respect to the time of D-Clas— versus the difference in the classification accuracy for each feature selection method and classifier. The high speed up values (from 50 to almost 700) show that the time required to find the threshold in "D-Comp" was lower than in our first approach D-Clas. Moreover, there was no a significant degradation in classification accuracy. In fact, in some cases the accuracy was improved (points located along the positive X axis).

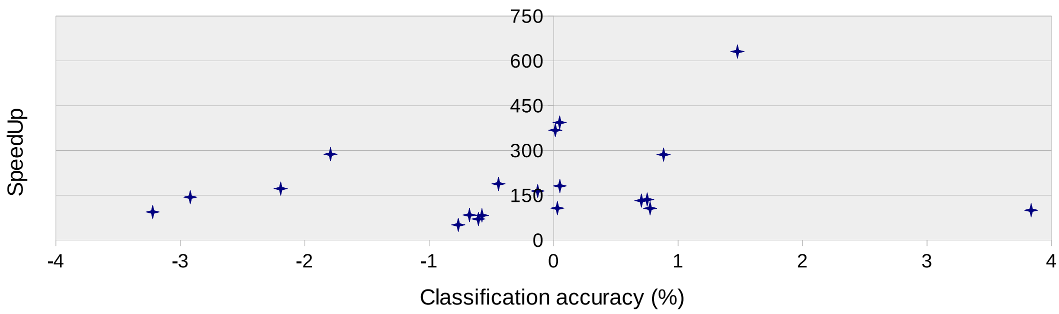


Figure 4.1: SpeedUp vs. classification accuracy achieved by the distributed approaches using a classifier for the threshold versus using a data complexity measure.

Bearing in mind that, in our proposed approach based on the Fisher's multiple discriminant ratio, runtime was significantly shortened, in this work we will analyze if there is still room for improving classification performance using other data complexity measures and different ratio samples/features.

4.3. Experimental settings

A description of the specific datasets used for this study—focusing on some of their particularities—is provided in this section, as well as the data complexity measures, classification algorithms and feature selection methods.

Datasets. In order to evaluate empirically the proposed distributed framework DFS-CM, we used 11 datasets, described in Table 4.1 in terms of the number of features, training and test samples and classes. For the horizontal distributed approach we employed the first six datasets. These datasets can be considered representative of medium to large problems, since the horizontal distribution is not clearly suitable for small datasets. For the vertical distributed approach we used three of the six datasets employed for the horizontal approach—those that have the highest number of features (Isolet, Madelon and Mnist)—and, moreover, five microarray datasets due to their high dimensionality. Those datasets originally divided into training and test sets were maintained, whereas, for the sake of comparison, datasets with only training sets were divided maintaining the class distribution using the rule 2/3 for training and 1/3 for testing.

Table 4.1: Characteristics of 11 datasets.

Dataset	#Features	#Samples		#Classes	Download
		Training	Test		
Connect4	42	45,038	22,519	3	[8]
Isolet	617	6238	1559	26	[8]
Madelon	500	1600	800	2	[8]
Ozone	72	1691	845	2	[8]
Spambase	57	3067	1534	2	[8]
Mnist	717	40,000	20,000	2	[8]
Breast	24,481	78	19	2	[2]
Gli85	22,283	56	29	2	[122]
CLL-SUB-111	11,340	74	37	3	[122]
Lung cancer	12,600	136	68	5	[115]
11-Tumors	12,534	114	58	11	[115]

Data complexity measures. Apart from the Fisher discriminant ratio (F1), two new measures are used for calculating the threshold of votes in our proposed method

DFS-CM, extended from the original definitions [10], so as to adapt them for datasets with multiple classes. The first one is F2 (Eq. 2.3 in Chapter 2). In the original definition of this measure [10], the authors compute the product (volume of the overlap region) instead of the sum (amount or size of the overlap region). The side effect of employing the product is that the value of this measure decreases drastically as dimensionality increases. Thus, as this can be a problem when dealing with high input dimensionality datasets, such as microarrays, we opted for using the sum. For multiclass problems, we compute F2 for each pair of classes, then obtain the absolute value for all of them, and finally return the sum of all these values. A low value of this measure means that the features can discriminate the instances of different classes. The second measure, N2, is the ratio of average intra/inter class nearest neighbor distance.

Table 4.2 reports the computational cost of each data complexity measure used in this work, where m is the number of samples, n the number of features and c the number of classes.

Table 4.2: Computational cost of the complexity measures.

Label	Data complexity measure	Computational cost
F1	Fisher discriminant ratio	$\mathcal{O}(m \cdot n \cdot c)$
F2	Length of the overlapping region	$\mathcal{O}(m \cdot n \cdot c)$
N2	Ratio of average intra/inter class nearest neighbor distance	$\mathcal{O}(m^2 \cdot n \cdot c)$

Classifiers. Four classifiers, each belonging to a different family, were chosen to evaluate the performance of the framework: two linear (naive Bayes and Support Vector Machine using a linear kernel) and two non-linear (C4.5 and 1-Nearest Neighbor). All four classifiers are executed in the Weka [54] environment, using default values for the parameters.

Feature selection methods. Feature selection methods have received a great deal of attention in the classification literature [19], which largely reflects filter, wrapper and embedded methods. Since wrapper and embedded method interactions with the classifier are computationally burdensome, we opted for filter methods as they have the advantage of being less computationally costly. In this work, five well-known filters were chosen: CFS, CONS, INT, IG and ReliefF; a description of both methods can be found in Appendix A.

While three of them return a feature subset (CFS, CONS and INTERACT), the other two (IG and ReliefF) are ranker methods, so a threshold is mandatory in order to obtain a subset of features. In this work we have opted for retaining the c top features, being c the number of features selected by CFS, since it is a widely used method and, among the three subset methods chosen, it is the one which usually selects the greatest number of features.

Table 4.3 shows the theoretical complexity of the five methods described above, where m is the number of samples and n is the number of features.

Table 4.3: Computational cost of the five feature selection methods focus of this work.

Method	Computational cost
CFS	$\mathcal{O}(m \cdot n^2)$
CONS	$\mathcal{O}(m \cdot n)$
INT	$\mathcal{O}(m \cdot n^2)$
IG	$\mathcal{O}(m \cdot n)$
ReliefF	$\mathcal{O}(m^2 \cdot n)$

4.4. Experimental results

In this section we present and discuss experimental results in terms of number of selected features, classification accuracy and runtime. Two different techniques for partitioning the data have been employed: horizontally (by samples) and vertically (by features). For each of these strategies, four different approaches will be compared: the centralized approach (C) and three distributed approaches (D-F1, D-F2 and D-N2) based on the data complexity measures Fisher's multiple discriminant ratio (F1), length of the overlapping region (F2) and ratio of average intra/inter class nearest neighbor distance (N2), respectively.

4.4.1. Horizontal partitioning

In order to test the proposed distributed method DFS-CM with horizontal partition, we selected six datasets which can be consulted in Table 4.1: Connect4, Isolet, Madelon, Ozone, Spambase and Mnist. The number of packets in which each training dataset is partitioned was calculated trying to maintain a proportion between the number of samples and the number of features, with the constraint of having, at least, three packets per dataset. According to this rule, the number of packets in which the dataset was divided was 3 for Madelon, 5 for Isolet, Spambase and Mnist, 11 for Ozone and 45 for Connect4.

Prior to any comparison, a study of the best alpha value for each distributed method based on a particular data complexity measure is provided. Tables 4.4 and 4.5 show the number of selected features and the classification accuracy achieved by each filter over the six datasets for the α values 0.25, 0.5 and 0.75. As can be seen, the number of selected features was smaller when α was 0.25. However, the best classification performances were obtained with $\alpha = 0.75$.

Table 4.4: Number of features selected by the horizontal distributed approaches with different alpha values.

	D-F1			D-F2			D-N2		
α	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
CFS	36.83	37.17	37.50	26.13	27.13	27.33	66.20	68.60	66.80
INT	24.00	25.33	27.17	32.17	33.83	34.17	40.20	42.40	44.4
Cons	17.50	17.83	19.00	15.00	17.33	17.33	22.20	21.20	22.00
IG	40.83	40.17	40.33	34.83	35.83	36.17	50.40	50.40	51.40
ReliefF	43.67	44.83	43.17	36.33	36.83	36.83	66.20	53.60	54.00
Average	**32.57**	33.07	33.43	**28.89**	30.19	30.37	**46.08**	47.24	47.76

In light of the results, although no significant differences were observed between the three values of alpha, this parameter was set to $\alpha = 0.75$, giving more influence to the value of the data complexity measure. This is because a lesser error is more important than a smaller storage requirement when directly selecting a subset of features.

Table 4.5: Classification accuracy achieved by the horizontal distributed approaches with different alpha values.

	D-F1			D-F2			D-N2		
α	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
CFS	76.48	76.36	76.60	70.03	71.83	72.56	73.77	73.96	74.00
INT	75.14	77.17	76.26	75.67	76.37	76.37	72.55	72.21	72.72
Cons	75.78	71.40	77.54	75.33	75.33	75.97	71.43	71.01	71.04
IG	77.36	78.25	76.56	74.38	74.93	76.93	74.52	74.39	74.60
ReliefF	76.17	77.11	77.13	73.83	74.46	74.67	72.84	74.47	74.43
Average	76.19	76.28	**76.82**	73.85	74.59	**75.30**	73.02	73.20	**73.36**

4.4.1.1. Number of selected features

Figure 4.2 displays the average of features selected by the five filters both with the centralized and distributed approaches. As can be seen, there were no significant differences between the number of features selected by centralized and distributed approaches, in some cases being even larger in the centralized approach (Isolet, Madelon and Ozone). Therefore, we can affirm that applying a distributed approach does not imply a larger selection of features.

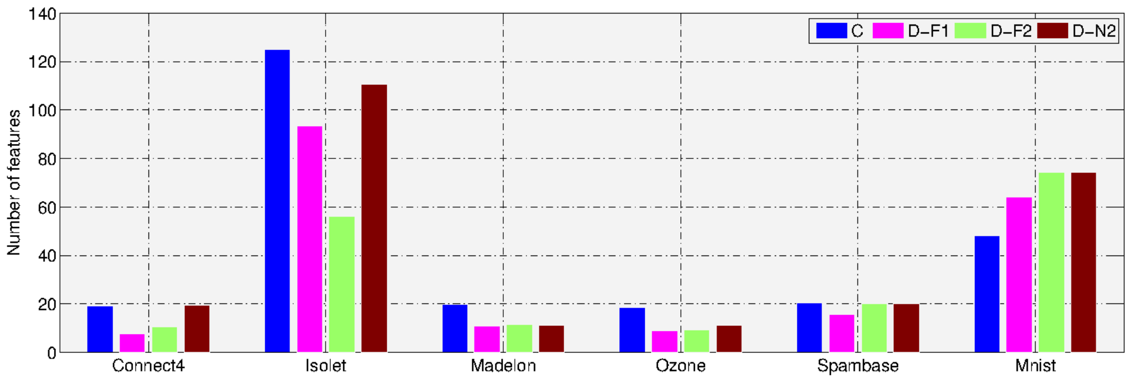


Figure 4.2: Comparing the centralized and distributed approaches using horizontal partition in terms of number of selected features.

4.4.1.2. Classification accuracy

In terms of classification accuracy, Tables B.5 and B.6 show the results obtained by the algorithms C4.5, NB, kNN and SVM both with the centralized and distributed approaches based on data complexity measures, whilst the best result for each dataset and approach is presented in Figure 4.3. As expected, the results were very variable depending on the dataset and classifier. For some datasets (Connect4 and Isolet) the highest accuracies were achieved by the centralized approach. For Spambase and Mnist the best results were obtained by our distributed approaches D-F2 and D-F1, respectively, whilst for Madelon and Ozone the best classification accuracy was the same for the four approaches.

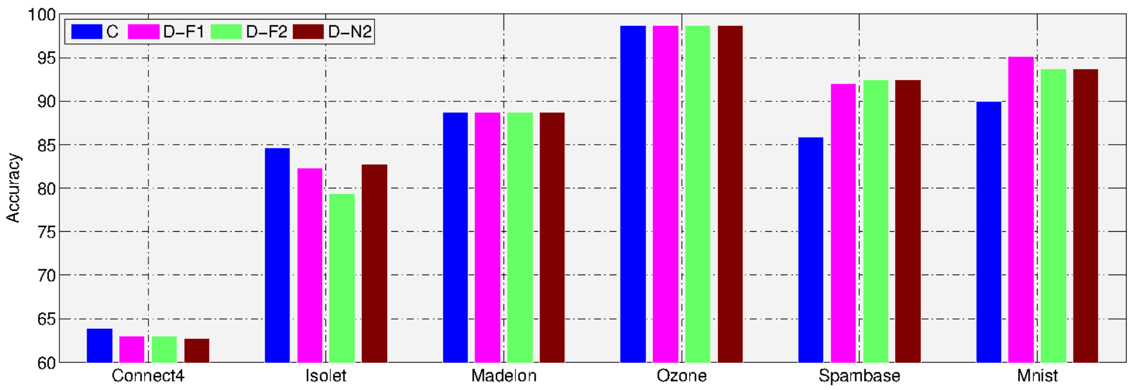


Figure 4.3: Comparing the centralized and distributed approaches using horizontal partition in terms of classification accuracy.

Following with the analysis, we studied the relationship between classifiers and data complexity measures. Figure 4.4 shows the average classification accuracy over the six datasets and the five feature selection methods. As expected, the best classification performance for the algorithm kNN was achieved by the D-N2 distributed approach. This is caused by the fact that the N2 measure was based in the nearest neighbor rule. For the rest of classifiers, the best classification accuracy was always achieved by the D-F1 approach.

In Figure 4.5, the classification accuracy achieved for a particular classification algorithm and feature selection method is shown. The NB and kNN classifiers obtained the best results for CONS. Regarding the ReliefF filter, C4.5 was the

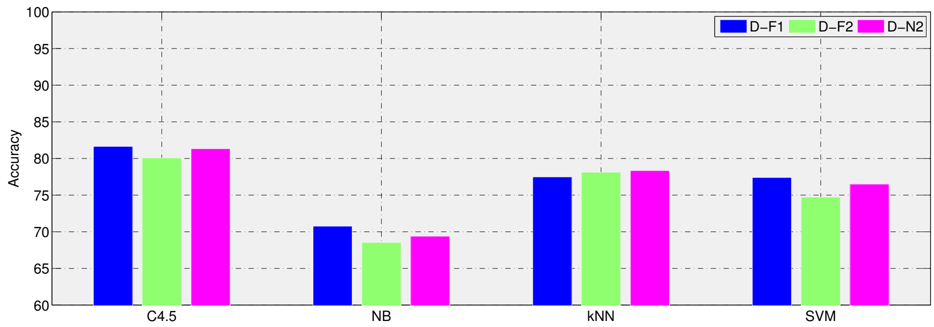


Figure 4.4: Relationship between the three distributed approaches and the four classifiers.

classifier which achieved the best classification performance whilst SVM was the best after applying IG.

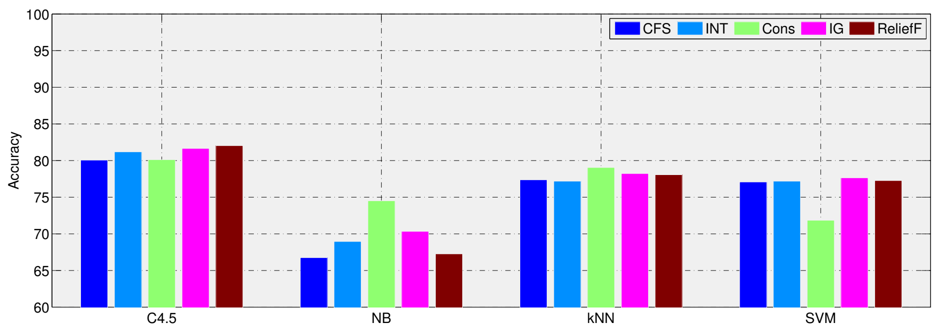


Figure 4.5: Relationship between the four classifiers and the five feature selection methods.

Figure 4.6 shows the relationship between the three distributed approaches and the five feature selection methods in terms of classification accuracy. As can be seen, the Cons and Relief filters were better with respect to the D-F1 whilst CFS and IG achieved the best results with the D-N2 distributed approach. In the case of the INT filter, this was the best feature selection method with respect to D-F2 approach. However, there were no significant differences between the classification performances achieved by the filters and the three distributed approaches based on data complexity measures.

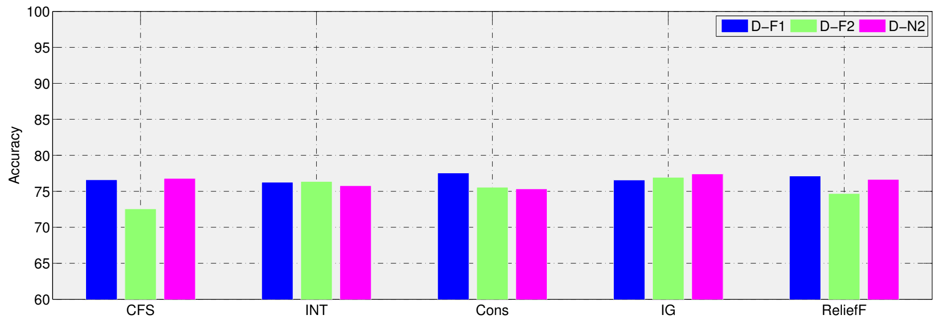


Figure 4.6: Relationship between the five feature selection methods and the three distributed approaches.

4.4.1.3. Runtime

Finally, Table 4.6 shows the runtime required by the feature selection methods, as well as the speed up values, which indicate the performance improvement of the DFS-CM distributed approach with respect to the centralized approach. Note that in the three distributed approaches, the feature selection stage at each packet was the same, so the time required will be referred as D for three of them. Also, in the distributed approach, given that all the subsets can be processed in parallel, the time displayed is the maximum of the times required by the filter in each subset generated in the partitioning stage.

As expected, the time required by the distributed methods was drastically reduced for all datasets and filters compared with that of the centralized approach, except for Ozone with the IG filter. It is worth mentioning the important reductions when the dimensionality of the dataset grew. For the Mnist dataset, which has 717 features and 40000 training samples, the reduction was more than notable. This fact proves the adequacy of the distributed approach when dealing with large datasets.

Furthermore, for the distributed approaches, it is necessary to take into account the time required to calculate the optimal threshold of votes to combine the partial results obtained on the different partitions of the data. In Table 4.7 we can see the average runtime on all datasets for each filter and distributed approach. Note that the time required to find the threshold in D-F1 and D-F2 was noticeably lower than

Table 4.6: Maximum runtime (s) for the feature selection methods tested. C stands for centralized approaches, while D refers to the distributed approaches.

		Connect4	Isolet	Madelon	Ozone	Spambase	Mnist	SpeedUp
CFS	C	100	250	36	10	12	1787	**5.73**
	D	10	77	25	8	6	257	
INT	C	112	196	40	9	13	3145	**10.56**
	D	11	70	31	8	14	199	
Cons	C	268	245	52	11	14	6163	**21.10**
	D	10	80	25	6	2	197	
IG	C	97	171	41	9	11	1451	**5.30**
	D	4	54	29	9	5	235	
ReliefF	C	1680	553	62	14	21	30413	**21.66**
	D	11	103	40	8	4	1346	

the one in D-N2. This was happening because the computational cost of the data complexity measure used by this method is higher than the other two (see Table 4.2).

Table 4.7: Average runtime (s) for obtaining the threshold of votes.

Method	D-F1	D-F2	D-N2
CFS	0.51	0.19	3605.14
INT	0.29	0.11	3559.31
Cons	0.16	0.07	3217.27
IG	0.28	0.11	3112.32
ReliefF	0.31	0.13	3281.38

4.4.2. Vertical partitioning

This section presents the results over eight of the datasets described in Table 4.1: Isolet, Madelon, Mnist, Breast, Gli85, CLL-SUB-111, Lung cancer and 11-Tumors. In the case of the first three datasets, we opted for partitioning them in five packets, so that each packet contained 20% of features. For the microarray datasets, the data was split by assigning groups of k features to each subset, where the number of features k in each subset was half the number of the samples, to avoid over-fitting. In this manner, the considered datasets had enough features to ensure correct learning.

Prior to any comparison, a study of the best alpha value for each distributed method based on a particular data complexity measure is provided. Tables 4.4 and 4.5 show the number of selected features and the classification accuracy achieved by each filter over the eight datasets, respectively. As can be seen, the best classification performances were obtained with $\alpha = 0.75$. Although the difference in the number of selected features is higher than in the case of the horizontal distribution, this parameter was set to 0.75, giving more influence to the value of the data complexity measure. This is because a better classification performance was more important than a smaller storage requirement.

Table 4.8: Number of features selected by the vertical distributed approaches with different alpha values.

	D-F1			D-F2			D-N2		
	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
CFS	974.50	981.00	1063.25	962.75	970.30	977.88	978.63	978.63	1098.00
INT	740.13	741.75	920.50	707.30	724.88	736.25	737.00	744.13	1109.50
Cons	265.38	264.63	312.50	251.88	268.88	268.88	265.25	292.00	295.00
IG	1051.88	1060.25	1116.00	989.30	989.30	1051.30	1053.25	1055.38	1176.5
ReliefF	802.50	797.13	805.00	749.25	784.3	796.75	911.63	911.63	911.63
Average	**766.88**	768.95	843.45	**732.10**	747.53	766.21	**789.15**	796.35	918.23

4.4.2.1. Number of selected features

As we can see in Figure 4.7, contrary to the horizontal distribution case, the number of features selected by distributed methods was larger than those selected by the centralized approaches. This is caused by the fact that, with the vertical partition, the features were distributed across the packets and it was more difficult to detect redundancy between features if they were in different partitions. Even so, our distributed approaches were using—in the worst case—33.68% (Isolet), 3.96% (Madelon), 8.81% (Mnist), 5.53% (Breast), 8.86% (Gli85), 6.43% (CLL-SUB-111),

Table 4.9: Classification accuracy achieved by the vertical distributed approaches with different alpha values.

	D-F1			D-F2			D-N2		
	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
CFS	78.34	77.93	77.77	76.75	76.79	77.09	77.44	77.44	76.71
INT	77.02	76.46	77.55	75.81	76.44	76.44	76.81	76.81	76.88
Cons	73.24	72.84	76.75	73.01	73.31	74.08	75.29	76.60	77.39
IG	77.63	77.39	78.04	76.26	76.27	76.93	77.76	77.76	77.60
ReliefF	78.53	78.15	78.09	77.52	78.75	78.31	78.57	78.57	79.21
Average	76.95	76.56	**77.64**	75.87	76.31	**76.60**	77.17	77.44	**77.56**

12.20% (Lung cancer) and 13.61% (11-Tumors) of the total features for the datasets chosen.

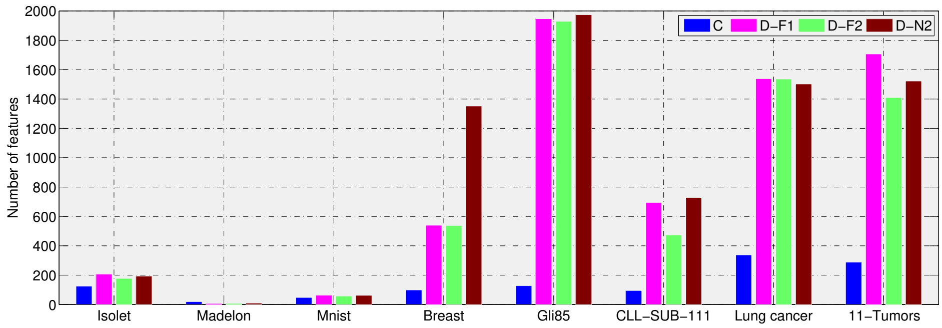


Figure 4.7: Comparing the centralized and distributed approaches using vertical partition in terms of number of selected features.

4.4.2.2. Classification accuracy

In terms of classification accuracy, Tables B.7 and B.8 show the best result for each dataset and classifier in bold face, whilst the best result for each dataset and approach is presented in Figure 4.8. As can be seen, in general there were no differences in terms of accuracy. As expected only the Gli85 dataset achieved the highest accuracy by the centralized approach. For 11-Tumors, the distributed approaches D-F2 and D-N2 obtained the same classification performance as the centralized approach. The best results for the rest of datasets were reported by one or several of the distributed approaches.

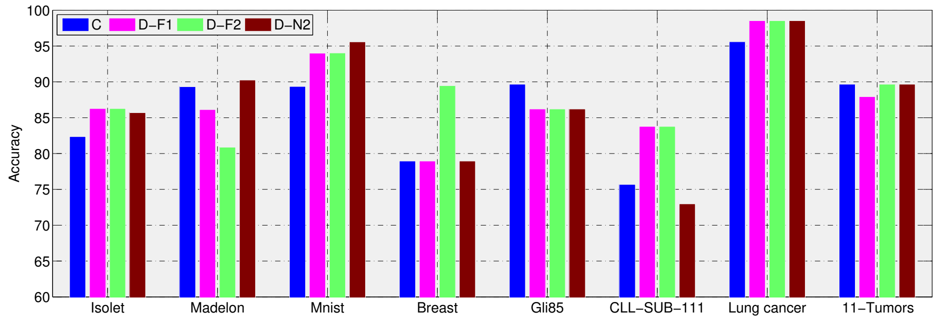


Figure 4.8: Comparing the centralized and distributed approaches using vertical partition in terms of classification accuracy.

Figure 4.9 shows the average classification accuracy over the eight datasets and the five filters for a particular classifier and data complexity measure. As happened in the horizontal partitioning, the best classification performance for the kNN algorithm was achieved by the D-N2 distributed approach, due to the fact that the N2 measure was based on the nearest neighbor rule. In Figure 4.10, the classification accuracy achieved for a particular classification algorithm and feature selection method is shown. The C4.5 and SVM classifiers obtained the best results for ReliefF. Regarding the IG filter, kNN was the classifier which achieved the best classification performance whilst NB was the best after applying INTERACT.

Figure 4.11 shows the correlation between the three distributed approaches and the five feature selection methods in terms of classification accuracy. As can be seen, the CFS, INT and IG filters are better with respect to the distributed approach using F1 as the data complexity measure. On the other hand, Cons and ReliefF achieved the best results with the D-N2 distributed approach.

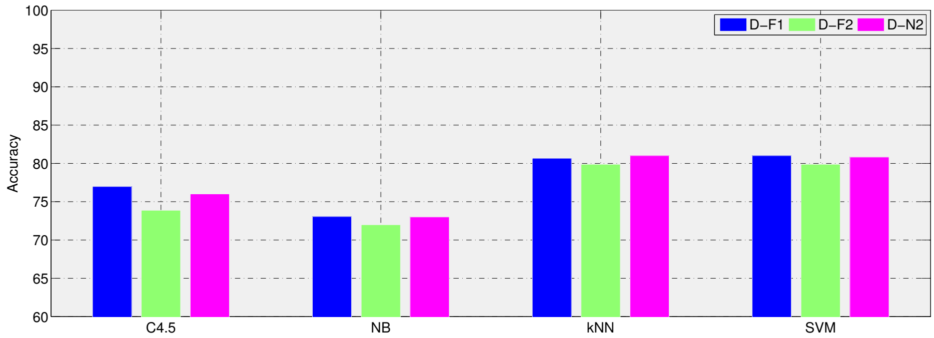


Figure 4.9: Relationship between the three distributed approaches and the four classifiers.

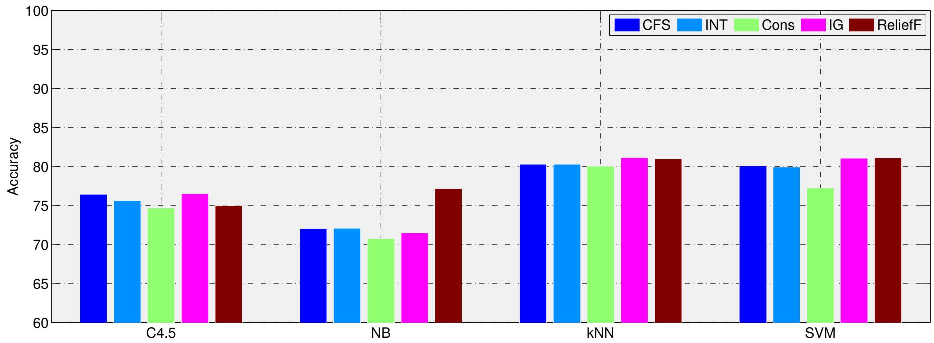


Figure 4.10: Relationship between the four classifiers and the five feature selection methods.

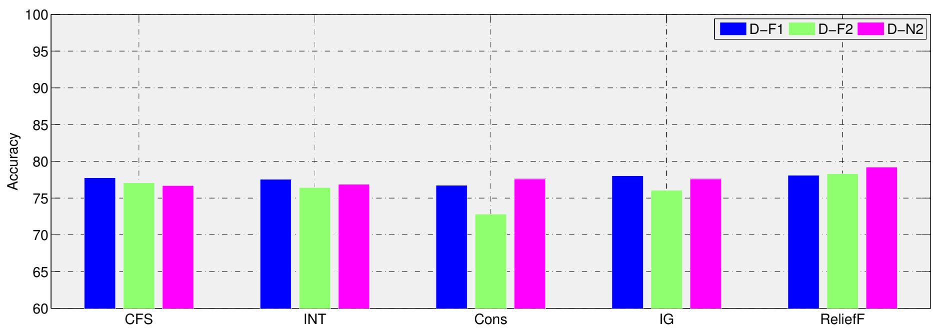


Figure 4.11: Relationship between the the five feature selection methods and the three distributed methods.

4.4.2.3. Runtime

Finally, Table 4.10 shows the runtime required by the feature selection methods, as well as the speed up values. In the case of the distributed approach, the time displayed is the maximum time among those obtained in the different packets. As happened with the horizontal distribution, this filtering time was independent of the classifier chosen. Again, the execution time was drastically shortened by applying the distributed approach, in some cases from 9434 seconds to 0.29 seconds (Lung dataset with CFS filter).

Table 4.10: Maximum runtime (s) for the feature selection methods tested. C stands for centralized approaches, while D refers to the distributed approaches.

		Isolet	Madelon	Mnist	Breast	Gli85	CLL-SUB-111	Lung cancer	11-Tumors	SpeedUp
CFS	C	250	36	1787	7969	7652	1335	9434	7959	**104.81**
	D	40	18	287	0.47	1.32	0.26	0.29	0.27	
INT	C	196	40	3145	179	121	45	135	79	**13.52**
	D	46	16	225	0.68	1.76	0.59	1.12	0.31	
Cons	C	368	52	6163	14.35	10.29	6.36	8.11	9.31	**21.98**
	D	58	16	225	0.33	1.33	0.29	0.45	0.33	
IG	C	171	41	1451	1.88	1.44	1.11	1.99	1.89	**5.03**
	D	42	15	273	0.61	0.93	0.21	0.24	0.22	
ReliefF	C	533	62	30413	3.95	2.31	1.65	5.05	5.49	**5.40**
	D	190	26	5522	0.70	1.76	0.24	0.26	0.27	

For the distributed approaches, it is necessary to take into account the time required to calculate the threshold to combine the partial outputs of features. Table 4.11 depicts the average runtime on all datasets for each filter and distributed approach. As happened with the horizontal distribution, the time required for the D-N2 approach increased considerably due to the high number of samples of some datasets. This becomes clear from examining the runtimes required by this approach for each filter and dataset in Table 4.12. As can be seen, the times were noticeably longer for the datasets with higher number of samples—specially Mnist, which has 40000 training samples—while for microarrays datasets these times were quite similar to those obtained by the D-F1 and D-F2 approaches.

In light of the results, we can conclude that the distributed approaches performed successfully—for both horizontal and vertical data partitioning—since the runtime was considerably reduced for the D-F1 and D-F2 approaches, while accuracy was

Table 4.11: Average runtime (s) for obtaining the threshold of votes.

Method	D-F1	D-F2	D-N2
CFS	3.12	0.96	3941.5
INT	2.48	0.74	4415.3
Cons	1.52	0.45	2916.5
IG	3.14	0.95	4464.6
ReliefF	3.16	0.96	3135.3

Table 4.12: Runtime (s) for obtaining the threshold of votes by the D-N2 approach.

Method	Isolet	Madelon	Mnist	Breast	Gli85	CLL-SUB-111	Lung cancer	11-Tumors
CFS	652.78	25.42	3084.20	0.77	1.02	0.79	6.67	2.34
INT	368.50	25.31	3491.78	0.78	1.11	0.78	6.19	2.31
Cons	181.76	25.32	2311.81	0.72	0.85	0.75	2.64	1.87
IG	644.52	25.37	2703.43	0.79	1.09	0.96	7.11	2.36
ReliefF	644.34	22.70	2440.24	0.80	1.24	0.90	7.22	2.38

maintained or even improved in some cases with regard to the standard centralized features selection process.

4.5. Case studies

Before discussing and analyzing the experimental results in detail, we will describe several case studies in order to extract some recommendations on the appropriate methods to use in certain specific scenarios: a) centralized vs. distributed approach, b) horizontal partitioning vs. vertical partitioning, c) distributed approaches based on three different data complexity measures, d) vertical distributed approach with no-disjoint partitions and e) experiments using a dataset which has approximately the same size in both dimensions.

4.5.1. Case study I: Centralized vs. distributed approach

Feature selection methods have been applied traditionally in a centralized manner. However, over the last few years many distributed methods have been developed. In this case study, we compare the centralized approach and the distributed approaches (average between the horizontal and vertical strategies) over 11 datasets in terms of number of selected features, classification accuracy and runtime. As we can see in Table 4.13, the number of features selected by the distributed methods was larger than those selected by the centralized approaches. In terms of classification accuracy, the best results were reported by the distributed approach, improving in 2% the centralized approach. Finally, the average maximum runtime (in seconds) for the feature selection methods tested on all datasets is illustrated in Table 4.13, as well as the speed up values, which indicate the performance improvement of the distributed with respect to the centralized approach. As expected, the advantage of the distributed approaches in terms of execution time over the standard method was significant. The time was reduced for all filters, showing a high speed up value. In light of these results, the authors recommend the use of the distributed approach.

Table 4.13: Comparison between centralized (C) and distributed (D) approaches.

	#Features		Accuracy (%)		Runtime (s)		SpeedUp
Method	C	D	C	D	C	D	
CFS	144.09	541.41	75.80	76.25	3322.18	34.42	96.52
INT	100.18	479.04	74.60	76.68	370.36	30.09	12.31
Cons	13.64	154.45	70.16	76.18	618.31	29.20	21.17
IG	144.09	575.72	75.59	77.24	162.57	31.29	5.19
ReliefF	144.09	439.17	76.09	77.35	2978.31	330.88	9.01
Average	**109.22**	437.96	74.45	**76.74**	1490.35	**91.18**	16.35

4.5.2. Case study II: Horizontal partitioning vs. vertical partitioning

As was stated above, two different ways to partition the original dataset were applied. In the horizontal partitioning (H), the dataset was divided into several packets that had the same features as the original dataset, each containing a subset of the original samples. In the case of vertical partitioning (V), the original dataset was divided into several packets that had the same number of samples as the original dataset, each containing a subset of the original set of features. In order to compare these approaches, the three common datasets used in Section 4.4 for the experimental setup were chosen: Isolet, Madelon and Mnist. Table 4.14 shows the results of these two techniques in terms of number of features, classification accuracy and runtime. As can be seen, the number of features selected by the distributed approach with vertical partition was noticeably larger than in the case of the horizontal distribution. This was happening because, with the vertical partition, the features were distributed across the packets and it was more difficult to detect redundancy between them. In terms of classification accuracy, the results were variable depending on the filter applied, though the best average classification accuracy on the three datasets and five filters was obtained by the vertical distributed approach. Finally, the time required by the horizontally distributed approach was smaller for all filters if compared with that of the distributed approach for vertical partitioning.

Table 4.14: Horizontal partitioning (H) vs. Vertical partitioning (V) in terms of number of features, classification accuracy and runtime.

	#Features		Accuracy (%)		Runtime (s)	
Method	H	V	H	V	H	V
CFS	58.22	99.00	75.27	76.59	1830	3500
INT	58.11	51.44	76.25	75.52	1520	3930
Cons	22.55	18.00	72.80	72.59	860	2590
IG	59.89	137.56	76.11	76.88	1210	3080
ReliefF	66.67	119.00	74.25	75.83	1110	2790
Average	**53.09**	85.00	74.94	**75.48**	**1310**	3180

In light of the results, if it is possible to select the way of partitioning the data, since the difference in classification accuracy was low (in the order of 0.7%) but the difference in terms of number of selected features and runtime was much bigger, the horizontal partition is preferable.

4.5.3. Case study III: Distributed approaches based on data complexity measures

Trying to overcome the issues of our first approach [21]—high runtimes to calculate the threshold of votes and methodology dependent of the classifier chosen—we propose new distributed approaches based on three different complexity measures. In Table 4.15 one can see a comparison among the three (D-F1, D-F2 and D-N2) in terms of number of selected features, classification accuracy and runtime. As can be seen, D-F1 obtained better accuracy by selecting a larger number of features than D-F2, which always selected insufficient features. On the other hand, D-N2 selected a number of features similar to D-F1, but with poorer classification performances in overall thus indicating that the selected subset by D-N2 was suboptimal. For each distributed approach based on a different data complexity measure, the average time required to build the final subset of features for all datasets is shown. Note that the time was notably less for the D-F2 and D-F1 approaches than for D-N2. As was explained in Section 4.4.1, this was happening due to the computational cost of the N2 complexity measure used by this method (see Table 4.2). In light of the above, the authors suggest using the Fisher discriminant ratio (F1) if classification accuracy is more important than reducing the storage requirements and runtime. In other cases, F2 is recommended.

Table 4.15: Comparison between D-F1, D-F2 and D-N2 approaches.

	#Features			Accuracy (%)			Runtime (s)		
Method	D-F1	D-F2	D-N2	D-F1	D-F2	D-N2	D-F1	D-F2	D-N2
CFS	1606.3	977.8	1098	77.18	74.82	76.01	1.81	0.57	3777.32
INT	920.5	676.8	1109.5	77.30	76.40	76.64	1.39	0.42	3987.31
Cons	312.5	268.8	295.5	77.15	75.02	76.84	0.84	0.26	3066.88
IG	1116	1051.3	1176.5	77.30	76.93	77.28	1.71	0.53	3788.46
ReliefF	805	796.8	911.6	77.62	76.49	77.69	1.73	0.54	3208.34
Average	952.1	**754.3**	918.2	**77.31**	75.93	76.22	1.49	**0.46**	3565.64

In the particular case of microarray data, as we mentioned in Section 4.4.2.3 (see Table 4.12), the times for the D-N2 approach were quite similar to those obtained by the D-F1 and D-F2 approaches. As the runtime is small for the three distributed approaches, it could be interesting to use the data complexity measure with the best classification accuracy (F1), although F2 shows competitive results in terms of number of selected features and runtime.

Table 4.16: Comparison between D-F1, D-F2 and D-N2 approaches on microarray datasets.

	#Features			Accuracy (%)			Runtime (s)		
Method	D-F1	D-F2	D-N2	D-F1	D-F2	D-N2	D-F1	D-F2	D-N2
CFS	1599	1505	1697.2	77.72	77.98	76.94	3.10	0.82	2.32
INT	1427	1154.6	174.02	78.83	77.74	76.89	2.89	0.78	2.23
Cons	484.8	421.6	450.2	78.49	77.51	77.70	2.04	0.56	1.37
IG	1704	1599	1801.8	79.55	80.47	80.47	3.14	0.82	2.46
ReliefF	1211.8	1205.8	1389.8	78.59	78.01	77.95	3.14	0.83	2.51
Average	1285.3	**1177.2**	1415.8	**78.59**	78.01	77.95	2.86	**0.76**	2.18

Following with the analysis, a study of the relationship between particular classifiers and data complexity measures was carried out. Figures 4.4 (horizontal partitioning) and 4.9 (vertical partitioning) shows the average classification accuracy for each classifier and distributed approach based on the three data complexity measures. As expected, the best classification accuracy for the algorithm kNN was achieved by the D-N2 distributed approach. This is caused by the fact that the N2 measure was based on distances to the nearest neighbor.

4.5.4. Case study IV: Distributed approach by features with no-disjoint partitions

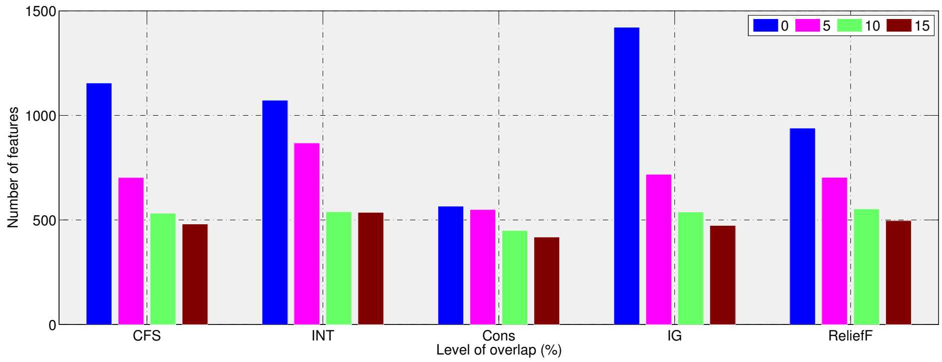
As was stated in Section 4.2, with the vertical partition, the first step of our method was to divide the training dataset into a number of disjoint packets of features (see line 3 in Algorithm 2). Thereby, the features were distributed across the packets and it was more difficult to detect redundancy between features. Trying to alleviate the problem of fighting redundancy among features, we propose using non-disjoint partitions, including a certain level of overlap in the packets. The overlap is defined as the subset of features that are common across different partitions.

In this case study, we consider three levels of overlapping (5%, 10% and 15%) and the Fisher discriminant ratio (F1) as the data complexity measure to establish the threshold of votes because of the results showed in Section 4.5.3. Figure 4.12 compares the results in terms of number of selected features and classification accuracy for the three approaches with overlap—one for each level—and the common vertical approach using disjoint packets of features (0%). As expected, a larger level of overlap decreased the number of selected features. However, in terms of classification accuracy, the distributed approaches with a certain level of overlap did not improve the results obtained with disjoint packets of features. This may be happening because the redundancy was positive, i.e. noise reduction and consequently better class separation may be obtained by adding features that are presumably redundant. Features that are independently and identically distributed are not truly redundant [51]. Thus, users should carefully balance both, number of features and classification performance, in order to decide if a small reduction in performance (in our experiments, of 1.25%), could be accepted for the sake of a more important reduction in the number of features (in our experiments 50%).

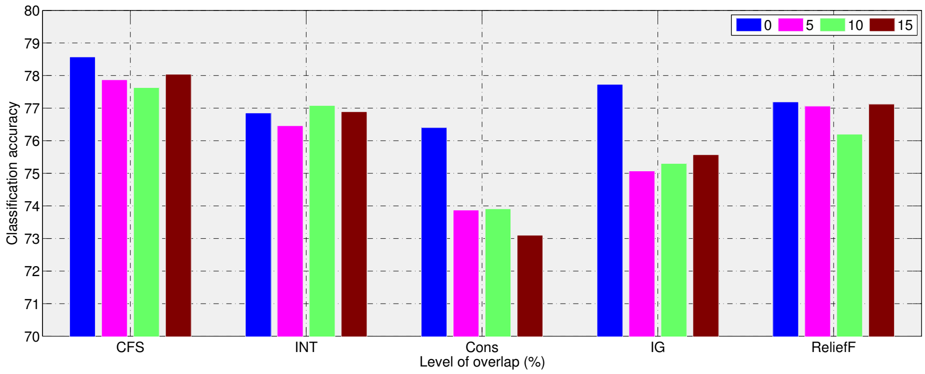
4.5.5. Case study V: Gisette dataset

The experiments carried out in this work involve some large datasets, in which either the number of examples or features are quite large. In this case study, we complete the experiments using a dataset that has approximately the same size in both dimensions, the Gisette dataset [52], which has 6000 samples (4000 for training and 2000 for test) and 5000 features. We considered both distributed approaches (horizontal and vertical partitioning) and the Fisher discriminant ratio and length of the overlapping region as data complexity measures to establish the threshold of votes. The number of packets into which the training dataset was divided in each round was four in the horizontal partitioning and three in the vertical partitioning.

Table 4.17 shows the results of these two techniques for the Gisette dataset in terms of number of selected features, classification accuracy and runtime (highest results highlighted in bold). The runtime was divided into two: the maximum runtime for the feature selection method tested plus the runtime for obtaining the threshold of votes.



(a) Number of features



(b) Classification accuracy (%)

Figure 4.12: Number of selected features and classification accuracy for the distributed approach D-F1 with different levels of overlap (5%, 10% and 15%).

In contrast to the results showed in case study II (see Section 4.5.3), the average number of features selected by the vertical distributed approach was smaller than in the case of horizontal partitioning, except for the ranker methods IG and ReliefF, which selected less features with the horizontal distribution. In terms of classification accuracy, the filters selecting more features in any distributed approach obtained the best classification performances. The runtime was variable depending on the filter applied, as expected, due to their different complexities (see Table 4.3). Whilst the times for CONS and ReliefF were higher when the vertical distributed approach was applied for CFS, INT and IG times were faster. Regarding the two data complexity measures used in the distributed approaches, F1 achieved the best classification performances (more significant in the vertical distribution) whilst F2 selected a lower number of features.

Table 4.17: Comparison between the distributed approaches on Gisette dataset. H stands for horizontal distribution approach, while V refers to the vertical distributed approach. Runtime is in seconds.

			Method					
			CFS	INT	Cons	IG	ReliefF	Average
#Features	H	D-F1	132	135	61	79	92	99.80
		D-F2	135	128	57	80	90	**98**
	V	D-F1	75	23	13	133	120	72.8
		D-F2	64	22	14	127	120	**69.4**
Accuracy	H	D-F1	94.31	94.14	94.36	86.56	92.68	**92.41**
		D-F2	94.10	93.94	93.91	86.75	91.71	92.08
	V	D-F1	92.50	89.50	73.74	88.81	89.91	**92.75**
		D-F2	92.31	86.70	86.34	88.51	93.15	89.40
Runtime	H	D-F1	191.33+0.04	155.56+0.02	89.40+0.02	2.75+0.03	93.40+0.02	**102.66+0.02**
		D-F2	191.33+0.02	155.56+0.02	89.40+0.01	2.75+0.02	93.40+0.02	**102.66+0.02**
	V	D-F1	15.98+0.09	24.85+0.11	110.48+0.05	2.43+0.10	175.03+0.10	65.75+0.09
		D-F2	15.98+0.06	24.85+0.05	110.48+0.06	2.43+0.06	175.03+0.07	**65.75+0.06**

The results of this study suggest the strategy of combining the partial results from the different partitions when the dimensions are equally big, partitioning the datasets by both instances and features together. In such a case, each subset will be formed by a subset of instances and a subset of features.

4.6. Summary

Nowadays, the standard approach to feature selection methods is centralized. As dataset sizes grow and models become more complex, it is natural to consider replacing centralized feature selection by parallel and distributed techniques, as a way to reduce costs. We argue that the existing feature selection algorithms may not work very well because of scalability and privacy concerns.

The main goal of this chapter was to design a method that would be able to successfully scale up feature selection algorithms, meaning that the runtime would be considerably reduced and the classification accuracy would do not drop to inadmissible values. Thereby, several methodologies for distributing the feature selection process based on data complexity measures have been proposed. First, we tackled the distribution commonly used in the literature: the horizontal partition. Then, we applied the same algorithm to data partitioned by features. The novel procedures proposed were able to reduce significantly the runtime while maintaining—or even improving—classification performance.

From the experiments carried out, we can draw some conclusions and make some recommendations to the users:

- **Which is the best approach to feature selection?**

  We have demonstrated that distributed methods show competitive results both in terms of runtime and classification accuracy. With respect to runtime, the behavior is excellent, being this fact the most important advantage of our method. Thereby, we recommend to use the distributed approaches instead of the traditional centralized methods when dealing with large datasets.

- **Which is the best way to partition data?**

  The partition of the datasets can be carried out in two ways: horizontally (by samples) or vertically (by features). In terms of runtime, this decision should be closely related to the complexity of the feature selection algorithm chosen. Depending on the design, the complexity of a certain feature selection algorithm may depend more on the number of features or on the number of samples of the dataset. That is the reason why it would be highly advisable to divide datasets by one or the other, depending on which factor determines the complexity of the algorithm. Due to the challenge of detecting redundancy between features in the vertical partition, the number of selected features is noticeably larger than in the case of the horizontal distribution. In contrast, the classification accuracy is better in the case of the vertical approach. Thus, if reducing the storage requirements and runtime is more important than classification accuracy, the horizontal partition is preferable.

- **Which is the best data complexity measure to calculate the threshold of votes?**

  Regarding the three different data complexity measures—Fisher discriminant ratio, length of the overlapping region and ratio of average intra/inter class nearest neighbor distance—used to combine the partial outputs obtained from each partition of data in the distributed methods, we found that the Fisher discriminant ratio (D-F1) achieved the best classification performance whilst

D-F2 shows interesting results in terms of number of selected features and runtime. Thus, if reducing the storage requirements and runtime is more important than classification accuracy, F2 is preferable. If classification accuracy is more important, F1 is recommended.

- **Which is the behavior of the vertical distributed approach when including a certain level of overlap in the feature subsets?**

  In the original definition of our distributed feature selection framework for vertical data partitioning, the training dataset was divided into a number of disjoint packets of features. As a result, it was more difficult to detect redundancy between features as they were distributed across the packets. By including a certain level of overlap (5%, 10% and 15%) we were able to reduce by 50% the number of selected features, whilst in the worst case scenario the classification accuracy decreased from 76.4% to 73.1% (see the CONS filter in Figure 4.12(b)). However, in some cases, including overlap in the feature partitions is not possible due to privacy concerns. An example could be that in which a hospital and a public health body wish to collaborate in order to analyze outbreak of certain diseases without actually revealing their data, making it impossible to join all the data together and distribute it with overlapping.

  In light of the above, the authors suggest distributing the data including a certain level of overlap when possible, taking into account not only privacy concerns, but also balancing loss of accuracy and reduction of features.

Chapter 5

Feature selection under computational constraints

Finally, and unlike the distributed approach followed in the previous chapter, we propose a complexity reduction of feature selection process based on the current philosophy of Edge Computing. With the advent and standardization of wireless connectivity paradigms and the cost reduction of electronic components, the number and diversity of IoT devices has exploded over the last decade [103]. Wearable computing has made successful and significant forays in fitness domains, health care, fashion and entertainment, among other application areas. These devices are usually employed as local systems, and their fundamental requirements are to work with little computing power and small memories. However, these requirements become challenging since emerging computing devices are not just sensor devices: they must perform sophisticated computation, collect and aggregate data for propagation to the cloud, and respond in real time to user requests. These data must be fed on a machine learning system to analyze information and make decisions. Unfortunately, limitations in the computational capabilities of resource-scarce devices inhibit the implementation of the most current machine learning algorithms on them. Then, the data must be sent to a remote computational infrastructure. However, Edge Computing has arrived to allow us to simplify the learning task in this scenario.

Imagine a health wearable (Figure 5.1) which measures a high number of body parameters such as vital signs (electrocardiography, pulse, blood oxygen saturation, respiration, skin temperature, CO_2), body kinematics as well as sensorial, emotional and cognitive reactivity such as electrocardiography, posture, fall, movement, speed, acceleration or pressure. It is common that a large number of these features (i.e. body parameters) is not informative because they are either irrelevant or redundant with respect to a specific disease or health condition. Therefore, selecting the most relevant features could significantly improve disease prevention, diagnosis, treatment, disease management and rehabilitation, and help to discover personal patterns of interest. Feature selection arises from the need of determining the "best" subset of variables for a given problem. Features can be categorized in three ways: relevant, irrelevant and redundant [62]. As a result, selecting the relevant features and ignoring the irrelevant and redundant ones is advisable. The process of feature selection is typically performed on a machine using high numerical representation, i.e. double-precision floating point calculations (64 bits). Using a more powerful general purpose processor provides significant benefits in terms of speed and capability to solve more complex problems. But this capability does not come without cost; a conventional microprocessor can require a substantial amount of off-chip support hardware, memory, and often a complex operating system [71]. In contrast to up-to-date computers, these requirements are often not met by embedded systems, low energy computers or integrated solutions that need to optimize the used hardware resources. However, to the best of our knowledge, reduced-precision approaches have not been implemented yet in the area of feature selection. And portable embedded systems, though, call for new feature strategies and methods that are able to deal with big dimensionality.

Figure 5.1: Health wearable [30].

In this chapter, we investigate feature selection by considering the information theoretic measure of mutual information with reduced precision parameters. The mutual information measure is used due to its computational efficiency and simple interpretation. Therefore, we are able to provide a limited bit depth mutual information, and—through different feature selection methods based on this measure—experimentally achieve classification performances close to that of 64-bit representations for several real and synthetic datasets. Our reduced precision approach is designed to analyze user level data, i.e. on-board analysis for close-loop feedback. It performs the preprocessing step over private "small" data. Then, this anonymized data could become available in the cloud, by aggregation of personal data from different users, to obtain "big" data that can be processed by more powerful processors and/or distributed to experts for further analysis.

5.1. Background

The majority of the existing approaches available investigated the effect of reduced precision in neural networks [88]. Han et al. [57] presented an energy-efficient engine that performed inference on compressed deep neural networks and accelerated the resulting sparse matrix-vector multiplication with weight sharing. Hubara et al. [63] introduced a method to train Quantized Neural Networks (QNNs), i.e. neural networks with extremely low precision weights and activations at run-time. They found that QNNs achieved prediction accuracy comparable to their 32-bit counterparts. Benoit et al. [65] proposed a quantization scheme that relied only on integer arithmetic to approximate the floating-point computations in a neural network. The authors were inspired by the work of [50], which leverages low-precision fixed-point arithmetic to accelerate the training speed of convolutional neural networks. In the area of Bayesian networks, Tschiatschek and Pernkopf [119] considered online learning of these classifiers with reduced precision parameters in order to facilitate their utilization in computationally constrained platforms. All above mentioned authors demonstrated that their proposed reduced-precision algorithms achieved classification performances close to that of Bayesian networks classifiers with parameters learned by traditional algorithms using double-precision floating point representation.

5.2. Mutual information in feature selection

Mutual Information (MI) comes from the field of Information Theory and it is widely used in both machine learning and statistics. One of its main uses is feature selection methods, and in fully supervised data, the features X are ranked using this measure, and the ones finally selected are those having the highest mutual information with the class label Y. The mutual information is defined as the expected logarithm of a ratio:

$$I(X;Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \ln \frac{p(x,y)}{p(x)p(y)} \tag{5.1}$$

where $p(x,y) = Pr\{X = x, Y = y\}$ is the probability mass function of the joint distribution when the random variable X takes on the value x from its alphabet $\mathcal{X}$ and Y takes on $y \in \mathcal{Y}$, while $p(x) = Pr\{X = x\}$ and $p(y) = Pr\{Y = y\}$ are the probability mass functions of the marginal distributions. In this work, the function is calculated in natural logarithm, so returned units are "nats". In practice we have to estimate this from data. This can be done by using the sample (maximum-likelihood) estimates of the probabilities $\hat{p}$ and plug them in the Equation 5.1. This maximum likelihood estimator for the mutual information is consistent [95], and as a result we have:

$$I(X;Y) \approx \hat{I}(X;Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \hat{p}(x,y) \ln \frac{\hat{p}(x,y)}{\hat{p}(x)\hat{p}(y)} \tag{5.2}$$

In order to calculate this we need the estimated distributions $\hat{p}(x,y), \hat{p}(x)$, and $\hat{p}(y)$. The probability of any particular event $p(X = x)$ is estimated by maximum likelihood, the frequency of occurrence of an event $X = x$ divided by the total number of events.

An example. Let us consider a vector Y with 651 observations, in which the number of occurrences of an event $Y = y$ is 3. The probability $\hat{p}(y)$ will be:

$$\hat{p}(y) = \frac{3}{651} = 0.00460829493$$

which it is approximately zero. For real applications, it is not necessary to store all the decimal digits, which makes mutual information an interesting measure to explore reduced precision. Besides, as the embedded systems market matures, we will likely see a movement away from full mutual information (i.e. 64 bit-representation) to limited approaches using a lower number of bits.

Mutual information definition is useful within the context of feature selection because it gives a way to quantify the output vector. Thus, there exist in the literature several feature selection methods based on mutual information measures [11, 118, 96, 49]. Most methods define heuristic functionals to assess feature subsets combining definitions of relevant and redundant features. Among the different feature selection methods based on mutual information, we have chosen three to evaluate our limited bit depth mutual information approach: MIM (Mutual Information Maximisation) [72] due to its simplicity, JMI (Joint Mutual Information) [129] and mRMR (minimum Redundancy Maximum Relevance) multivariate filter [96] since they showed the best overall trade-off for accuracy/stability [26]a description of both methods can be found in Appendix A. In any case, our reduced precision approach could be easily implemented in any other MI-based feature selection algorithms.

Table 5.1 shows the theoretical complexity of the three methods described above [108]. Let us assume that we have a dataset of m samples and n features and we want to select the top-k.

Table 5.1: Theoretical complexity of the three feature selection methods focus of this work.

Method	Complexity
MIM	$\mathcal{O}(k \cdot m \cdot n)$
JMI	$\mathcal{O}(k^2 \cdot m \cdot n)$
mRMR	$\mathcal{O}(k^2 \cdot m \cdot n)$

5.3. Limited bit depth mutual information

In information theoretic feature selection, the main challenge is to estimate the mutual information, one of the most common measures of dependence used in machine learning. As said above, to calculate mutual information we need to estimate the probability distributions. Internally, it counts the occurrences of values within a particular group (i.e. its frequency). Thus, based on Tschiatschek and Pernkopf [119]'s work for approximately computing probabilities, we investigate mutual information with limited number of bits by considering this measure with reduced precision counters. To perform the reduced precision approach, we target a fixed-point representation instead of the 64-bit resolution used typically by the standard hardware platforms. Fixed-point numbers are essentially integers scaled by a constant factor, i.e. the fractional part has a fixed number of digits. We characterize fixed-point numbers by the number of integer bits bi and the number of fractional bits bf. The motivation to move to fixed-point arithmetic is two-fold. The first reason is that these bit representation compute units are typically faster and consume far less hardware resources and power than the conventional floating-point computations. And, second, low-precision data representation reduces the memory footprint, enabling larger models to fit within the given memory capacity and lowering the bandwidth requirements.

Mutual Information parameters are typically represented in the logarithm domain. For the reduced precision parameters, we compute the number of occurrences of an event and use a lookup table to determine the logarithm of the probability of a particular event. The lookup table is indexed in terms of number of occurrences of an event (individual counters) and the total number of events (total counter) and stores values for the logarithms in the desired reduced precision representation. To limit the maximum size of the lookup table and the bit-width required for the counters, we assumed some maximum integer number M. The lookup table L is pre-computed such that:

$$L(i,j) = \left[\frac{ln(i/j)}{q}\right]_R \cdot q \tag{5.3}$$

where $[\cdot]_R$ denotes rounding to the closest integer, q is the quantization interval of the desired fixed-point representation (2^{-bf}), $ln(\cdot)$ denotes the natural logarithm, and where the counters i and j are in the range $\{0,...,M-1\}$.

Given certain specific data, the individual counters c_j^i and the population C are computed according to Algorithm 3. Following the fixed-point representation, we assumed some maximum integer number M, where $M = 2^{(bf+bi)} - 1$ in terms of number of fractional bits bf and number of integer bits bi. After calculating the cumulative count C, we ensure that it is in range. Different from Tschiatschek's algorithm, we also divide by two the individual counters c_i when C reaches its maximum value (lines 9–12 in Algorithm 3). The problem we encountered with the original algorithm was that sometimes the total counter could be lower than the individual counter. And in order to estimate the mutual information, it gave us poor approximations of the logarithmic probabilities.

Algorithm 3 Our reduced precision algorithm for MI

1: **Require:** Individual counters c_j^i and total counter C; lookup table L
2: **for** i, j **do**
3:     **if** $c_j^i = M$ **then** ▷ maximum value reached?
4:         $c_j^i \leftarrow c_j^i/2 \ \forall i, j$ ▷ half counters (round down)
5:     **end if**
6: **end for**
7: $C = \sum (c_j^i)$ ▷ sum of the individual counters
8: **while** $C \leq M$ **do** ▷ ensure that C is in range
9:     $C \leftarrow C/2$
10:     $c_j^i \leftarrow c_j^i/2 \ \forall i, j$ ▷ revise index correction
11: **end while**
12: $l_j^i \leftarrow L(c_j^i, C) \ \forall i, j$ ▷ get the log-probability from lookup table
13: **return** l_j^i

5.3.1. Empirical study

Below we empirically evaluate our limited bit depth mutual information in terms of accuracy—using bias and variance measures—and ranking similarity over synthetic data.

5.3.1.1. Accuracy in terms of bias/variance

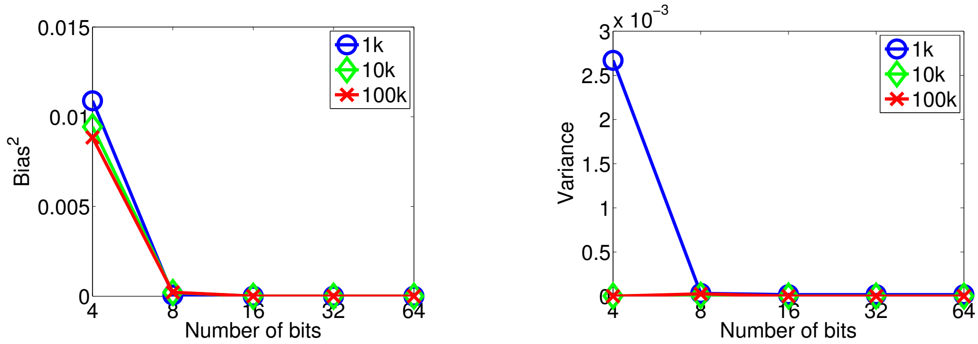
To evaluate the performance of the reduced precision mutual information against the full version using a 64-bit representation, we generate synthetic data with two different degrees of dependency with the target class Y. To create the data, firstly we generate the values of Y, by taking n samples from a Bernoulli distribution with $p(y = 1) = 0.5$. Then, we choose the parameters $p(x|y)$ that guarantee the desired degrees of dependency in terms of $I(X;Y)$ and we use these parameters to sample the values of X. All criteria need an estimate of the mutual information between a feature or a feature set and the class variable, which is derived from a finite dataset. For that reason, the accuracy of the estimator plays a crucial role in the ranking of features. The bias and variance are used to measure the accuracy, which can be defined as:

$$\begin{aligned} bias\left(\hat{I}(X;Y)\right) &= \mathbb{E}\left[\hat{I}(X;Y)\right] - I(X;Y) \\ var\left(\hat{I}(X;Y)\right) &= \mathbb{E}\left[\left(\hat{I}(X;Y) - \mathbb{E}\left[\hat{I}(X;Y)\right]\right)^2\right] \end{aligned}$$

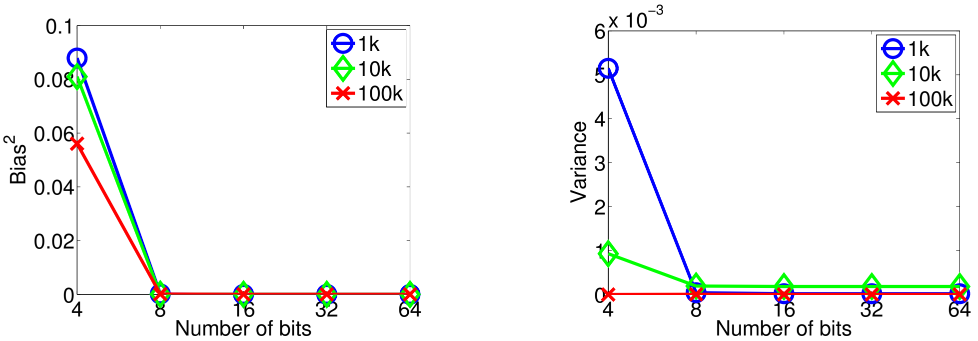
Figure 5.2 shows the results of an experimental study considering two different degrees of dependency, $I(X;Y) = 0.01$ and $I(X;Y) = 0.1$, and three sample sizes, 1k, 10k and 100k. Bias/variance is obtained for the different limited bit depth mutual information versions (4, 8, 16 and 32 bits) and the full mutual information (64 bits), which will be the baseline method for comparison. As can be observed, the bias for 8, 16 and 32 bits converges to the 64-bit representation. Besides, the reduced precision MI using 4 bits does not converge but it is consistent, since both bias and variance decrease as the sample size increases.

5.3.1.2. Similarity rankings

Our limited bit depth mutual information described above will be used within a feature selection procedure. The output of a feature selection algorithm might be: a scoring over the features, a ranking of the features or a feature subset. In this section, we aim at illustrating the performance of our limited bit mutual information in terms of feature ranking variability. Let us assume there are d features in total. A ranking r can be formed as a vector of d distinct natural numbers taken from 1 to d. To measure the similarity of the feature rankings obtained by the reduced precision mutual information with different number of bits, we use the Spearman rank-order correlation coefficient [15], also commonly called Spearman's ρ. This



(a) Small effect - $I(X;Y) = 0.01$



(b) Large effect - $I(X;Y) = 0.1$

Figure 5.2: Comparing the performance of our bit limited depth mutual information (4, 8, 16 and 32 bits) with the full mutual information (64 bits) in terms of bias^2/variance. To estimate bias/variance we average over 5000 runs. Please note different axes for the different variables Bias and Variance.

coefficient takes values in the range $[-1, 1]$, where 1 means that the two rankings are identical, -1 means that there is no correlation between them. To be able to do this, we need to know the "true" ranking [109]. For this task, we generate various synthetic datasets consisting of $d = 10$ and $d = 20$ features with different degrees of dependency with the target class Y in terms of mutual information. The mutual information $I(X, Y)$ population values for each feature are:

- "Easy" scenario with 10 features: [2 4 6 8 10 12 14 16 18 20] $\times 10^{-2}$.
- "Difficult" scenario with 20 features: [2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21] $\times 10^{-2}$.

where a high mutual information translates into a high rank of the feature. The arity of features is chosen randomly between the following values $|\mathcal{X}| = 2, 5, 10$ and 20. The experiment was repeated taking different sample sizes from 1000 to 100,000 samples to observe the performance when the sample size increases. To estimate the Spearman's ρ we average over 100 runs.

Table 5.2 shows the Spearman's ρ obtained for the different limited bit depth mutual information versions (4, 8, 16 and 32 bits) and the full mutual information (64 bits). The lower values of the reduced precision approach using 4 bits shows that the correlation between its ranking and the "true" ranking is quite poor in both scenarios. However, from 8 bits all the approaches achieved a Spearman's ρ coefficient close to 1, which means that the rankings obtained by these approaches are similar to the "true" rankings. Moreover, we can observe that by increasing the sample size all of the reduced precision approaches improve their rankings, and they are closer to the "true" ranking in both scenarios. However, differences between both scenarios can be seen when 16 and 32 bits are used. In the difficult scenario, these limited bit depth MI versions do not get the "true ranking". This could be because there is a smaller distance between the population values of the mutual information, and thus the ranking will change.

Table 5.2: Spearman's ρ coefficient

#Features	#Samples	#Bits				
		4	8	16	32	64
10	1000	0.215	0.963	0.983	0.983	0.983
	10,000	0.326	0.963	1.000	1.000	1.000
	100,000	0.429	0.974	1.000	1.000	1.000
20	1000	0.179	0.975	0.973	0.973	0.973
	10,000	0.320	0.973	0.995	0.995	0.995
	100,000	0.472	0.984	0.996	0.996	0.996

In light of the results obtained, we proceed to use our limited bit depth mutual information approach within a more sophisticated method. In this work, we have chosen to apply this to feature selection. Despite the poor results using 4 bits, we have kept this approach in order to see how it affects to the accuracy of feature selection methods.

5.4. Application in feature selection

Our bit limited depth mutual information described above can be applicable to any method that uses internally the mutual information measure. In this work, we have chosen to do so within feature selection since this process has a key role to play in helping reduce high-dimensionality in machine learning problems [19], and it is lately specially relevant with the advent of Big Data. There is a large number of feature selection methods that use mutual information as a measure, thus their performance depending on the accuracy obtained by the mutual information step. As mentioned before, we have chosen to implement our reduced precision approach in the MIM, JMI and mRMR filters methods due to their popularity and good results in the machine learning area, but analogous implementations could be derived for any other FS method based on mutual information. We have considered several synthetic—of which the relevant features are already known—and real datasets. Table 5.3 details the main characteristics of the chosen datasets: for each dataset, the number of features, the number of samples and the number of classes. Experiments were executed in the Matlab2018a and Weka environments.

Table 5.3: Characteristics of the datasets.

Dataset	Type	#Features	#Samples	#Classes
Arcene	Real	10,000	200	2
Congress	Real	16	435	2
Connect-4	Real	42	45,038	3
CorrAL-100	Synthetic	100	100,000	2
GISETTE	Synthetic	5000	6000	2
Led-500	Synthetic	500	200,000	10
Splice	Real	60	3175	3
Waveform	Real	40	5000	3

- **UCI** datasets [75]. This is a collection of datasets of which we have selected Arcene, Congress, Connect-4, Splice and Waveform, with small to medium number of samples. The features within each dataset have a variety of characteristics: some are binary/discrete, and some are continuous. Continuous features were discretized, using an equal-width strategy in 5 bins, while features already with a categorical range were left untouched.

- **GISETTE** is a handwritten digit recognition problem from the NIPS 2003 Feature Selection Challenge [52]. Features were discretized independently into 10 equal width bins.

- **CorrAL-100**. The CorrAL dataset [67] has six binary features (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 was constructed by adding 93 features irrelevant binary features to the previous CorrAL dataset. The data for the added features was generated randomly. 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 to the class label by 75%, so if one applies a classifier using only this feature, a 25% of error should be obtained.

- **LED-500**. The LED problem [25] is a simple classification task that consists of, given the active LEDs on a seven segments display, identifying the digit that the display is representing. Thus, the classification task to be solved is described by seven binary attributes and ten possible classes available. A 1 in a attribute indicates that the LED is active, and a 0 indicates it is not active. Led-500 was constructed by adding 493 irrelevant binary features.

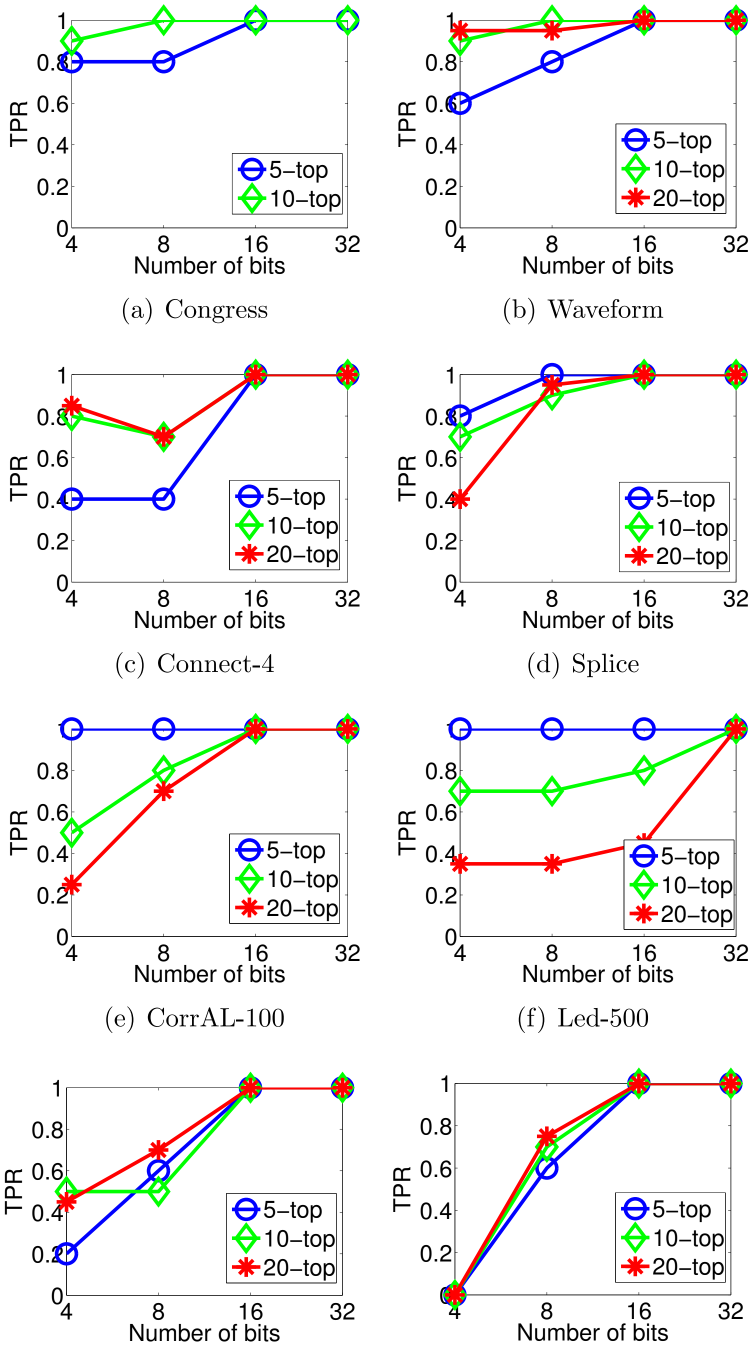
In the following sections we present and discuss the experimental results in terms of the quality of the selected features and the classification accuracy.

5.4.1. Quality of the selected features

To evaluate the similarity between the rankings obtained by the reduced precision versions and the 64-bit mutual information after performing the MIM, JMI and mRMR methods, we show the true positive rate for each dataset. The true positive rate measures the proportion of features that are correctly identified as such, using the full mutual information version (64 bits) as the ideal ranking. In high dimensional datasets, it is common to focus only on the top features, so in these experiments we compare only the k top features, with $k = 5, 10$ and 20 for all datasets except Congress, for which only the results with the 5 and 10 top features are shown as it has only 16 features.

Figures 5.3, 5.4 and 5.5 show the true positive rate (TPR) over the eight datasets presented in Table 5.3. The datasets are sorted in ascending order by their number of total features. As can be seen, for the datasets with less than 100 features—Congress, Waveform, Connect-4 and Splice—, our reduced precision approach using only 16 bits selected the same $5, 10$ and 20 features that the full version. Moreover, for the smaller datasets in terms of sample size, Congress and Splice, the reduced precision approach was able to achieve a 100% true positive rate even using 8 bits. When the number of features of the dataset increases, the performance of our reduced precision version using 16 bits started to decrease, and the same effect appears for datasets with high number of samples. The challenge with high dimensionality can be clearly seen in the Arcene dataset, where the limited bit depth MI using 4 bits does not select correctly any feature. For CorrAL-100 dataset, even the reduced precision version using only 4 bits was able to return the same 5 top features of the full version using 64 bits. It might be happening because this dataset has four relevant features (f_1, f_2, f_3 and f_4) and another feature that is correlated to the class label by 75%. This means that there is a slight difference between the mutual information values of these features and the rest of features. Therefore, we can say that in general, 16 bits are sufficient to select the same features that the full version using 64 bits.

Comparing the results between the different feature selection methods, we can see that JMI performs better—in some cases—than MIM and mRMR when 8 bits are used. This could be because JMI criterion has the best trade-off in terms of stability and flexibility over other feature selection methods based on Information Theory due to its nature (it balances the relevancy and redundancy terms and includes the conditional redundancy) [26].



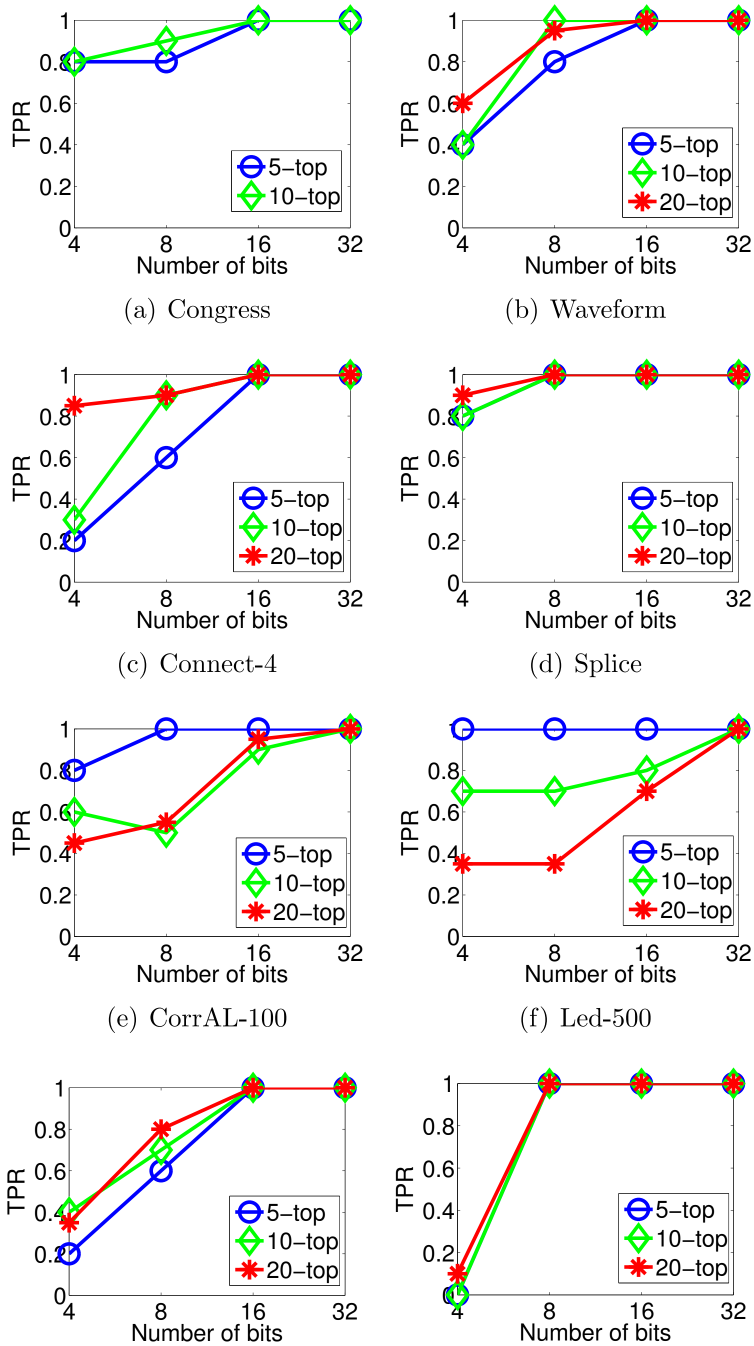
(a) Congress (b) Waveform

(c) Connect-4 (d) Splice

(e) CorrAL-100 (f) Led-500

(g) GISETTE (h) Arcene

Figure 5.3: TPR of the different reduced precision approaches using MIM.



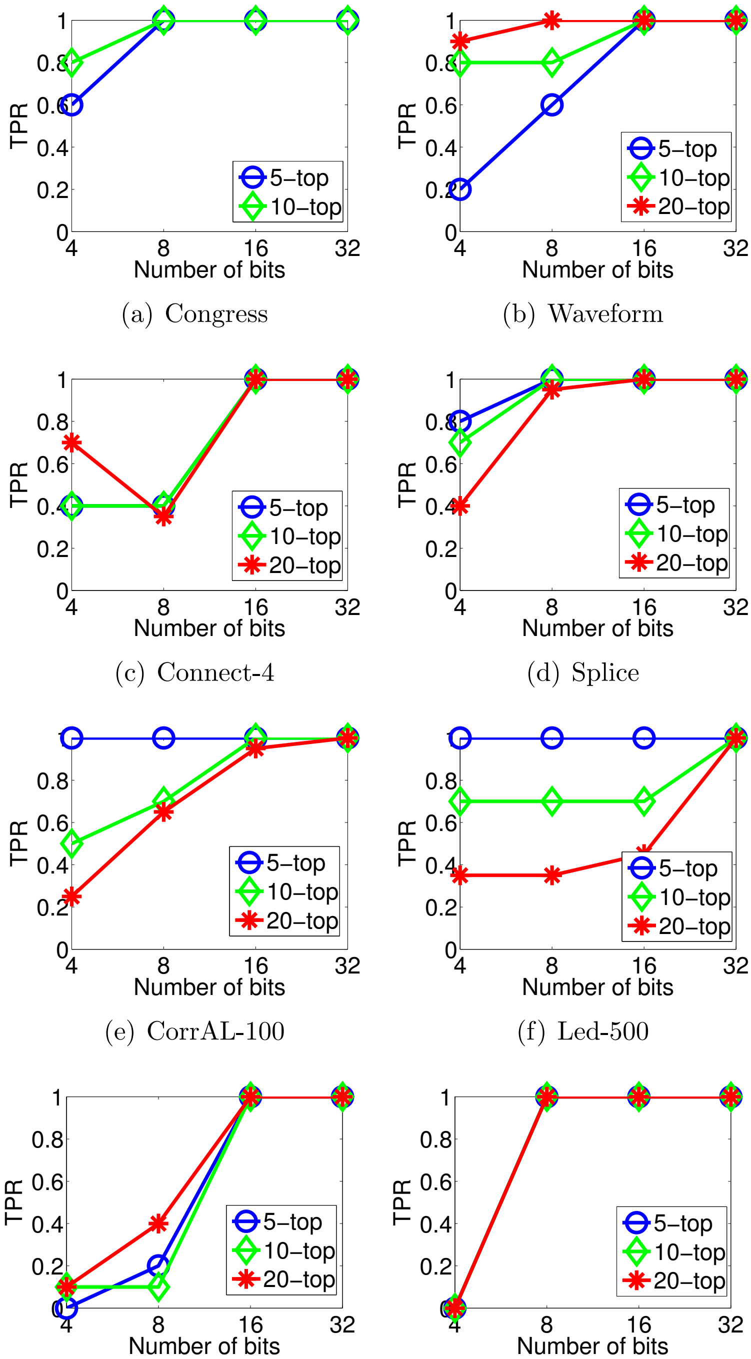
(a) Congress (b) Waveform

(c) Connect-4 (d) Splice

(e) CorrAL-100 (f) Led-500

(g) GISETTE (h) Arcene

Figure 5.4: TPR of the different reduced precision approaches using JMI.



(a) Congress (b) Waveform

(c) Connect-4 (d) Splice

(e) CorrAL-100 (f) Led-500

(g) GISETTE (h) Arcene

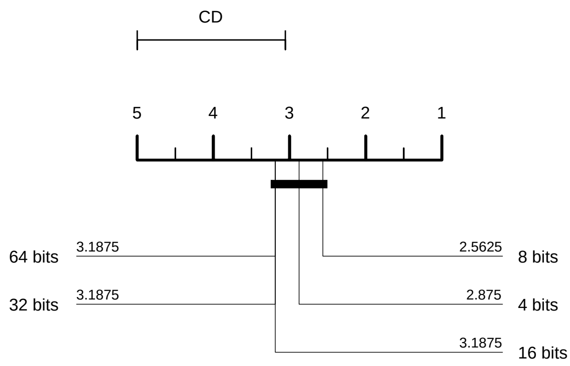
Figure 5.5: TPR of the different reduced precision approaches using mRMR.

5.4.2. Classification accuracy

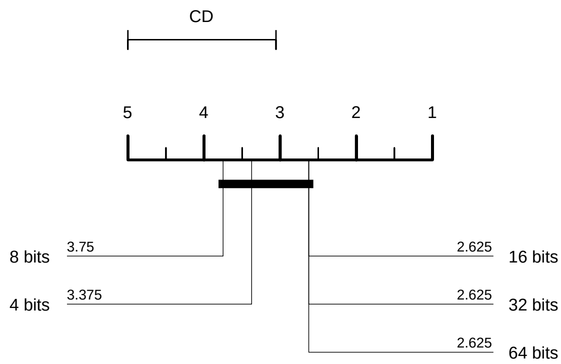
After the feature selection process, and in order to estimate whether the reduced precision in the feature selection process might affect classification, a study using classifiers was carried out. At this point, it is necessary to clarify that including classifiers in our experiments is likely to obscure the experimental observations related to feature selection performance using a limited number of bits, since they include their own assumptions and particularities. Therefore, in these experiments, we used a simple nearest neighbor algorithm (with number of neighbors $k = 3$) [3] as classifier since it makes few assumptions about the data, and we avoid the need for parameter tuning. To estimate the error rate we computed a 5-fold cross validation. For evaluating the performance of the reduced precision approaches, we compared the results obtained when using the ranking built with 4, 8, 16, 32 and 64 bits. Due to the large number of results, some tables have been moved to Appendix B.

To explore the statistical significance of our classification results, we analyzed the ranks of the reduced precision approaches by using a Friedman test with the Nemenyi post-hoc test. Figures 5.6, 5.7 and 5.8 present the critical difference diagrams, introduced by Demšar [37], where groups of methods that are not significantly different (at $\alpha = 0.10$) are connected. As can be seen for the three different k top selected features and JMI and mRMR methods, 64, 32 and 16 bits perform better on average but with no statistical significance over the reduced precisions approaches using only 4 and 8 bits, with the exception of mRMR (Figure 5.8). In the case of MIM, although there is no statistical significance over the reduced precisions approaches, the best performance is not always achieved through versions with 16, 32 or 64 bits. This could be because this last method assumes that each feature is independent of all other features. However, where features may be interdependent, this is known to be suboptimal. In general, it is widely accepted that a useful and parsimonious set of features should not only be individually relevant, but also should not be redundant with respect to each other [26].

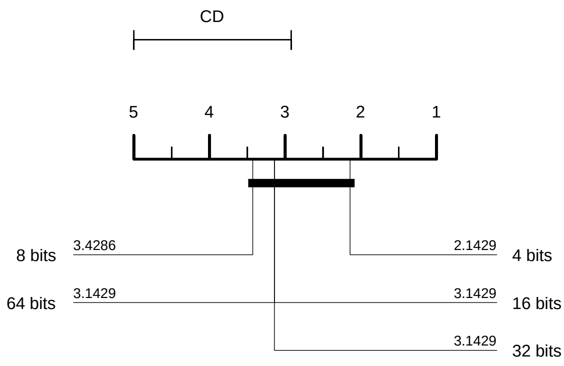
In summary, these experiments demonstrate that with a small number of bits the rankings change, but this variation does not affect significantly the classification accuracy, since this measure is the ultimate form of evaluation of the goodness of a feature ranking method.



(a) 5-top features

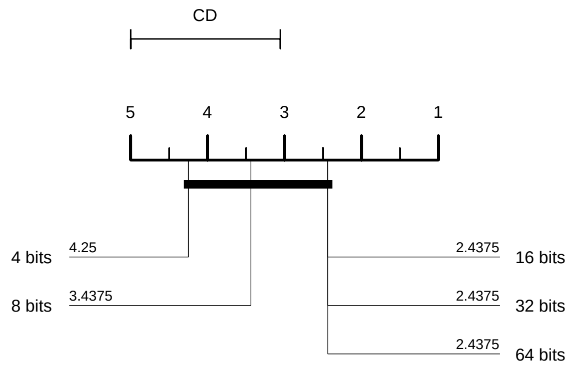


(b) 10-top features

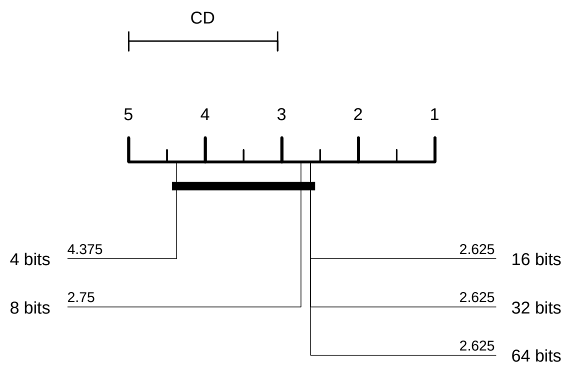


(c) 20-top features

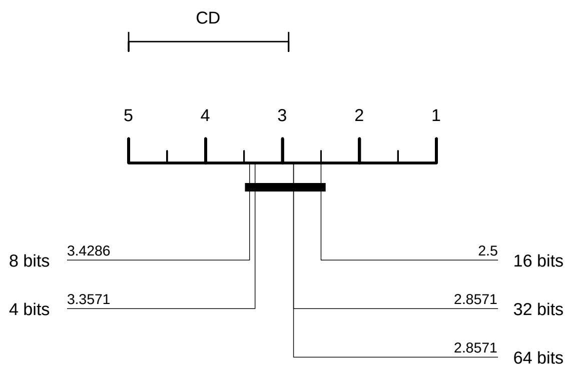
Figure 5.6: Critical difference diagrams showing the average ranks after applying MIM on the four reduced precision approaches (4, 8, 16 and 32 bits) and the full version (64 bits) for three different k-top selected features.



(a) 5-top features

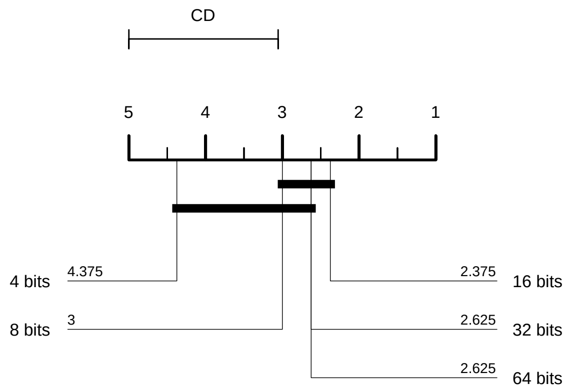


(b) 10-top features

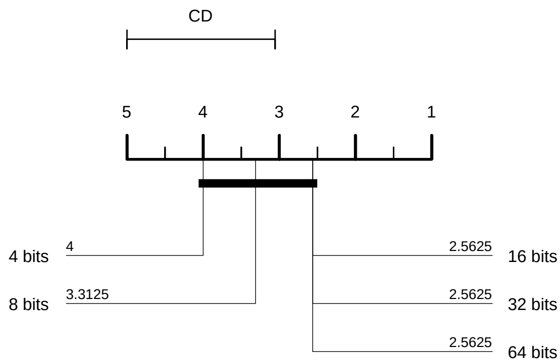


(c) 20-top features

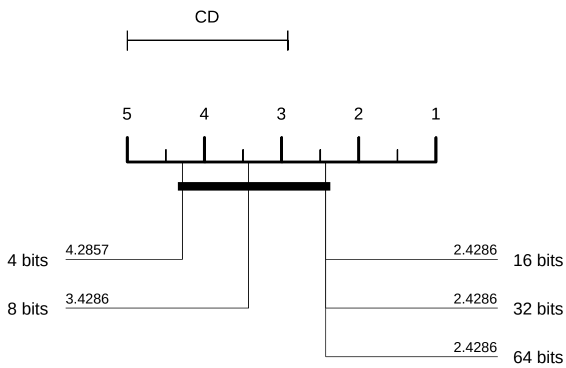
Figure 5.7: Critical difference diagrams showing the average ranks after applying JMI on the four reduced precision approaches (4, 8, 16 and 32 bits) and the full version (64 bits) for three different k-top selected features.



(a) 5-top features



(b) 10-top features



(c) 20-top features

Figure 5.8: Critical difference diagrams showing the average ranks after applying mRMR on the four reduced precision approaches (4, 8, 16 and 32 bits) and the full version (64 bits) for three different k-top selected features.

5.4.3. Case study: Dealing with noise in the inputs: LED

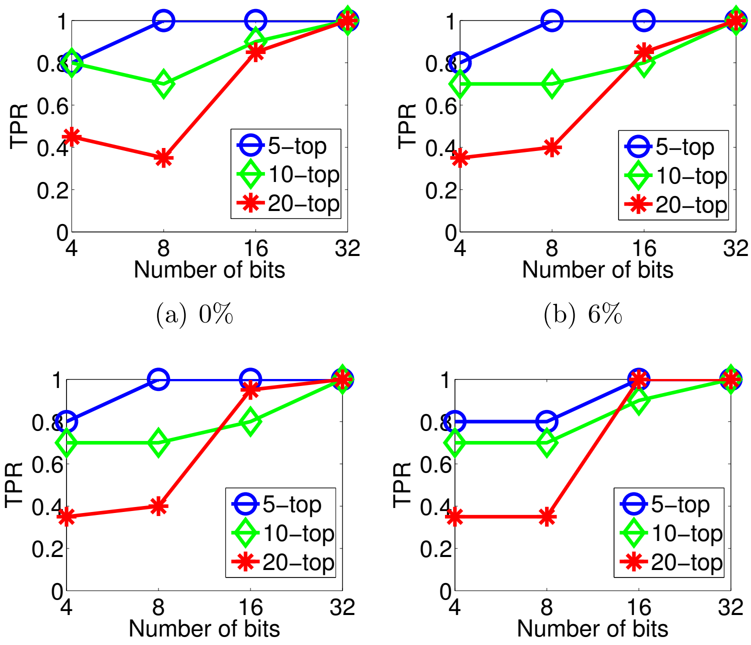
The LED dataset consists of correctly identifying seven LEDs that represent numbers between 0 and 9. Some irrelevant features were added forming the Led-500 dataset (493 irrelevant features). In order to make this dataset more complex, different levels of noise in the inputs (6%, 10% and 20%) were added [34]. In this manner, the tolerance to different levels of noise of the bit limited depth MI tested will be checked. Note that, as the attributes take binary values, adding noise means assigning to the relevant features an incorrect value. Besides, and unlike the Led-500 dataset used above, the number of samples was reduced to 10,000 so that its volume does not affect the study of noise.

In this case study, we consider JMI as the feature selection method due to the results obtained in Section 5.4.1. Figure 5.9 depicts detailed results of these experiments. It is interesting to note that the presence of noise does not seem to influence our limited bit depth MI, except in the case of 20% of noise and 5-top features. The reduced precision approach was not able to achieve a 100% true positive rate using just 8 bits.

With regard to the classification accuracy, it decreases as the level of noise increases, as expected (Table 5.4). However, the results using 4 bits are somewhat misleading. The 4-bit reduced precision version achieved better results in terms of classification accuracy than other versions with a larger number of bits. This is happening because—due to the high number of features of LED dataset and the low values of mutual information (99% under 0.006, see Figure 5.10)—the reduced precision approach does not get to sort the features by following the JMI criterion and it returns the feature ranking following its order in the original dataset. And, in this particular synthetic dataset, the classification task to be solved is described by the first seven binary attributes.

5.4.4. Comparison with baseline method

As we mentioned above, Tschiatschek and Pernkopf [119] proposed Bayesian Network classifiers when reducing the precision of the probability parameters. Since mutual information also needs to estimate probabilities, our work was built upon



(a) 0% (b) 6%

(c) 10% (d) 20%

Figure 5.9: TPR of the different reduced precision approaches using JMI over LED dataset with different levels of noise (6%, 10% and 20%).

Table 5.4: Classification accuracy (%) and standard deviation for LED dataset with different levels of noise (6%, 10% and 20%).

Top features	Noise (%)	#Bits 4	8	16	32	64
5	0	100.00 ± 0.00	89.67 ± 0.00	89.67 ± 0.00	89.67 ± 0.00	89.67 ± 0.00
	6	94.00 ± 0.00	84.16 ± 0.00	84.16 ± 0.00	84.16 ± 0.00	84.16 ± 0.00
	10	90.00 ± 0.01	81.10 ± 0.01	81.10 ± 0.01	81.10 ± 0.01	81.10 ± 0.01
	20	80.00 ± 0.01	64.52 ± 0.01	72.27 ± 0.01	72.27 ± 0.01	72.27 ± 0.01
10	0	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	6	94.00 ± 0.00	94.00 ± 0.00	94.00 ± 0.00	94.00 ± 0.00	94.00 ± 0.00
	10	90.00 ± 0.00	90.00 ± 0.00	90.00 ± 0.00	90.00 ± 0.00	90.00 ± 0.00
	20	80.00 ± 0.01	80.00 ± 0.01	80.00 ± 0.01	80.00 ± 0.01	80.00 ± 0.01
20	0	98.71 ± 0.00	98.93 ± 0.00	98.75 ± 0.00	98.79 ± 0.00	98.79 ± 0.00
	6	92.52 ± 0.00	92.18 ± 0.01	92.33 ± 0.01	92.23 ± 0.01	92.23 ± 0.01
	10	87.90 ± 0.00	87.85 ± 0.00	87.99 ± 0.01	87.84 ± 0.01	87.84 ± 0.01
	20	76.65 ± 0.00	76.48 ± 0.00	76.39 ± 0.00	76.39 ± 0.00	76.39 ± 0.00

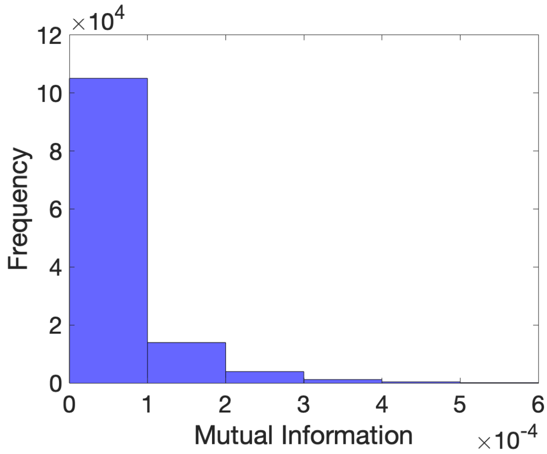


Figure 5.10: Histogram of frequency distribution values of mutual information of LED dataset.

this idea. In order to analyze Tschiatschek's algorithm on the mutual information measure, we generated synthetic data in the same way as in Section 5.3. The degree of dependence with the target class in terms of mutual information was fixed to 0.1 and the number of samples to 10,000. All criteria need an estimate of the mutual information between a feature or a feature set and the class variable, which is derived from finite dataset. For that reason, the accuracy of the estimator plays a crucial role in the ranking of features. To measure the accuracy we use the *Mean Square Error* (MSE), which is calculated from the ground truth we know from artificial data. To estimate MSE we averaged over 5000 runs. Both look up tables are calculated in natural logarithm, so the returned units are "nats".

Figure 5.11 compares Tschiatschek's algorithm with our proposal in terms of MSE. As can be seen, for the reduced precision approaches using only 4 and 8 bits, Tschiatschek's algorithm obtained high values of MSE while our proposed limited bit mutual information method achieved values close to zero. Besides, we can observe that with 16 and 32 bits both algorithms converge. It is necessary to clarify that Tschiatschek and Pernkopf [119] performed their experiments using 10 bits. As we aimed to explore the effect of the reduced precision with a small amount of bits, we redefined this algorithm with the aim of achieving better performance for our limited bit mutual information version.

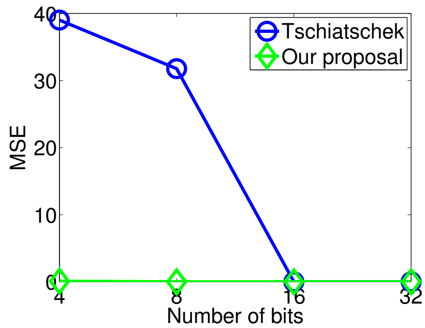


Figure 5.11: Comparing the performance of Tschiatschek's algorithm with our proposed reduced precision mutual information in terms of Mean Square Error, $I(X;Y) = 0.1$.

5.5. Summary

Since the development and commercialization of wearable technology is growing expansively, we have seen an opportunity to develop machine learning methods, specifically feature selection algorithms, in computationally constrained platforms. In this work we have proposed mutual information using reduced precision parameters within a feature selection procedure. Experimental results over several synthetic and real datasets have shown that 16 bits are sufficient to return the same feature ranking than that of 64-bit representation. As a result, meaningful benefits will be provided when implementing mutual information in embedded systems for on-device analysis. Having on device machine learning has some tremendous profits regarding privacy, reliability, efficient use of network bandwidth and power saving.

From the experiments carried out, we can draw some conclusions and make some recommendations to the users:

- Our reduced precision approach will not be adequate if there is a small distance between the population values of the mutual information. Besides, the ranking will be more unstable in the bottom of the list, where the features contain less and less information.
- When the number of features of the dataset increases, we will need more bits. Nevertheless, it is important to note that our reduced precision approach was designed to analyze user level data. If we are working in a big data scenario,

data is probably collected from different users, and so it would be processed either by more powerful central processors or distributed in different nodes by further analysis.

- Regarding the three feature selection methods used to test our limited bit depth mutual information, we have found that JMI was the most stable. However, if we take into account the computational cost of these methods, MIM seems to the most appropriate for this scenario.

- With respect to the presence of noise, it does not seem to influence appreciably our limited bit depth MI . In terms of classification accuracy, and as expected, it decreases as the level of noise increases.

Chapter 6

Conclusions and Future Work

Data is growing at an unprecedented pace. With the variety, changing nature and volume—both in number of samples and features—of data flowing through networks, databases and IoT devices, it has become more and more difficult to find patterns that lead to meaningful conclusions. At the same time, organizations need to find ways to obtain some value from all of this data. This thesis is devoted to reduce the complexity of machine learning methods in the current scenario.

In the present chapter we summarize the main contributions of the work carried out in this thesis and provide some insights of future research directions.

- **Data complexity.** Intrinsic characteristics extracted from the training datasets of classification problems have proven to be effective predictors. Among them, data complexity measures can be used to identify data particularities which imply some difficulty in separating the data points into their expected classes—such as the shape of the decision class boundary, the amount of overlap among the classes, the proximity of two classes or the number of informative samples available for training. This information can reduce the complexity of machine learning techniques, which can in turn be focused on challenges highlighted by such characteristics of the problems. Our experimental results for 21 microarray datasets demonstrated a correlation between several data complexity measures and classification performance, thereby enabling conclusions regarding the best classifiers for particular datasets. We also observed that feature selection and, specially the "Correlation-based Feature Selection" filter, reduced the complexity of the data.

Besides, the task of choosing the appropriate classifier for a problem is not an easy-to-solve question due to the high number of algorithms available belonging to different families. Most of these classification algorithms exhibit a degradation in the performance when faced with many irrelevant and/or redundant features. Thus, it would be interesting to analyze the impact of feature selection in classification. We have already performed some preliminary work in this research line, that will be published in an international conference (ESANN 2020). In this study, the initial experimental results over ten synthetic datasets showed that the significance of selecting a given classifier notably decreased after applying an appropriate preprocessing step. As future research, we plan to extend this study to other scenarios including real datasets, as well as analyzing the complexity reduction—through data complexity measures—after feature selection has been performed.

- **Distributed learning.** With the proliferation of large-scale data, researchers must focus not only on the accuracy but also on the scalability of the existing methods. Thus, a possible strategy is to distribute the learning task over a number of nodes. In this distributed scenario, class probabilities can be shown to be a weighted average of the individual class probabilities within each node. These weights depend on the marginal probabilities of the instance over each node and over the entire data set. In this work, two different approaches to approximate these weights were proposed. The first one was based on estimating the distance between feature distributions between each node and the test set, while the second one controlled the contribution of each instance of the test set in order to minimize these distributional distances. The resulting learning models exhibited interesting properties including that they worked with any local classifier and did not require retraining the classifiers or sharing information between individual nodes, thus conforming privacy-preserving methods. The experimental results on several real and synthetic datasets reported benefits in terms of classification accuracy, particularly when the second approach was considered. Besides, we assessed the "class imbalance change" problem.

  There are several topics related to our approach to be further considered. It will be interesting to check the usefulness of our approach to select one or a small subset of nodes to perform the classification and evaluating whether a significant degradation in accuracy is observed. Another future research direction involves the study of a linear-time approximation of the second approach.

- **Distributed feature selection.** The big data explosion now has the added problem of big dimensionality. Thus, similar to learning algorithms, we will have to deal with scability if we apply feature selection to this new scenario. The main goal was to distribute the feature selection process, expecting that the execution time would be considerably shortened and the accuracy would not degrade to poor values. Several proposals have been presented in Chapter 4, in an attempt to deal with both horizontal and vertical partitioning of the data. The experimental results on several datasets demonstrated important savings in runtime and satisfactory performance, since the classification accuracy matched and in some cases even improved the results of the original feature selection algorithms over the whole datasets.

  While the initial findings are promising, further research is necessary. As future research, we plan to partition the datasets by both instances and features together, developing new strategies to combine the partial results from the different partitions. We are considering possibilities for implementing this method on Big Data platforms such as Spark [6] or Flink [4].

- **Feature selection under computational constraints.** Different from the distributed approach followed in the two previous chapters, in Chapter 5 we focused on reducing the complexity of the feature selection task from the philosophy of Edge Computing. Due to the proliferation of mobile computing and Internet of Things devices, there is an urgent need to push the machine learning frontiers to the network edge so as to fully unleash the potential of the edge big data. Bearing this is mind, and the limitations in the computational capabilities of these devices, we have proposed mutual information using reduced precision parameters within a feature selection procedure. To test the adequacy of the proposed approach, we have implemented it in several well-known feature selection algorithms, applied to a broad suite of synthetic and real datasets. The obtained results demonstrated that low bit representations were sufficient to achieve performances close to that of double precision parameters and thus open the door for the use of feature selection in embedded platforms that minimize the energy consumption and carbon emissions. As future research, we plan to use our limited bit depth mutual information in a Markov Blanket context.

As can be seen, this thesis covers a broad suite of problems arisen from the advent of big data and the explosion of the Internet of Things. The proposed approaches have demonstrated their capability to reduce the complexity of traditional machine learning algorithms, and thus it is expected that the contribution of this thesis will open the doors to the development of new machine learning methods that are simpler, more robust, and more computationally efficient.

In addition to the future works that have been mentioned above, most of which are already in progress, there are some other lines of research that we would like to tackle. Another alternative solution to handle large-scale datasets is to apply prototype reduction techniques [89]. The goal of this technique is to build a reduced version of the training set that improves the accuracy of future predictions and speeds up the execution of algorithms. Thus, it would be interesting to apply the idea of measuring similarity between high dimensional distributions proposed in Chapter 3 in prototype reduction techniques.

Finally, we plan to use our limited bit depth mutual information as preprocessing step of some low precision classifiers since industry investment and research interest in edge computing have grown dramatically in recent years. A relevant example is that of the autonomous car, that it is estimated can generate 4TB of data per day, with only the cameras transmitting between 30-40 Mbits/s to the system, and adding the 10-100 Kbits/s of the radar. It is obvious that the car can not wait to communicate with the cloud to obtain an answer, it is indispensable for the own devices, and the car central computer to be able to analyze data—using both learning and preprocessing methods—and respond to the needs of autonomous driving at all times. Of course, the cloud will still remain important, as the information and knowledge derived can be transferred so that the rest of cars can benefit from the experience and results of the reaction to different events.

6.1. Publications from the thesis

The contents of the present research have been published in the following specialized journals, conferences and book chapters:

Journal publications

- L. Morán-Fernández, V. Bolón-Canedo and A. Alonso-Betanzos (2017). Centralized vs. distributed feature selection methods based on data complexity measures. Knowledge-Based Systems, 117, 27-45. JCR Q1.

- L. Morán-Fernández, V. Bolón-Canedo and A. Alonso-Betanzos (2017). Can classification performance be predicted by complexity measures? A study using microarray data. Knowledge and Information Systems, 51(3), 1067-1090. JCR Q2.

- P. Montero-Manso, L. Morán-Fernández, V. Bolón-Canedo, J. A. Vilar and A. Alonso-Betanzos (2019). Distributed classification based on distances between probability distributions in feature space. Information Sciences, 496, 431-450. JCR Q1.

Journal publications (Under review process)

- L. Morán-Fernández, K. Sechidis, V. Bolón-Canedo, A. Alonso-Betanzos and G. Brown (2018). Feature selection with limited bit depth mutual information for portable embedded systems. Knowledge-Based Systems.

International conferences

- V. Bolón-Canedo, L. Morán-Fernández and A. Alonso-Betanzos, A. (2015, July). An insight on complexity measures and classification in microarray data. In 2015 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

- L. Morán-Fernández, V. Bolón-Canedo, A. Alonso-Betanzos (2017, April). A distributed approach for classification using distance metrics. In European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN).
- L. Morán-Fernández, V. Bolón-Canedo, A. Alonso-Betanzos (2020, April). Do we need hundreds of classifiers or a good feature selection?. In European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN).

National Conferences

- L. Morán-Fernández, V. Bolón-Canedo and A. Alonso-Betanzos (2015, November). A time efficient approach for distributed feature selection partitioning by features. In Conference of the Spanish association for artificial intelligence (pp. 245-254). Springer, Cham.
- L. Morán-Fernández, V. Bolón-Canedo and A. Alonso-Betanzos, A. (2018, September). Feature selection with limited bit depth mutual information for embedded systems. Multidisciplinary Digital Publishing Institute Proceedings, 2(18), 1187.

Book chapters

- A. Alonso-Betanzos, V. Bolón-Canedo, C. Eiras-Franco, L. Morán-Fernández and B. Seijo-Pardo, B. (2018). Preprocessing in High Dimensional Datasets. In Advances in Biomedical Informatics (pp. 247-271). Springer, Cham.
- A. Alonso-Betanzos, V. Bolón-Canedo, L. Morán-Fernández and B. Seijo-Pardo (2019). Feature Selection Applied to Microarray Data. In Microarray Bioinformatics (pp. 123-152). Humana, New York, NY.
- A. Alonso-Betanzos, V. Bolón-Canedo, L. Morán-Fernández and N. Sánchez-Maroño (2019). A Review of Microarray Datasets: Where to Find Them and Specific Characteristics. In Microarray Bioinformatics (pp. 65-85). Humana, New York, NY.

Bibliography

[1] KDD Cup 99 dataset. [Online; accessed April-2017]. pages 60

[2] T. Agency for Sciency and Research. Kent ridge bio-medical dataset repository. [Online]. Available: http://leo.ugr.es/elvira/DBCRepository/. pages 27, 80

[3] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine learning*, 6(1):37–66, 1991. pages 121, 154

[4] Apache Software Foundation. Apache Flink. https://flink.apache.org/. pages 3, 133

[5] Apache Software Foundation. Apache Hadoop. https://hadoop.apache.org/. pages 3, 47

[6] Apache Software Foundation. Apache Spark. https://spark.apache.org/. pages 3, 47, 133

[7] Apache Software Foundation. MLlib. https://spark.apache.org/mllib/. pages 47

[8] K. Bache and M. Linchman. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences. [Online; accessed January 2016]. http://archive.ics.uci.edu/ml/. pages 80

[9] M. Banerjee and S. Chakravarty. Privacy preserving feature selection for distributed data using virtual dimension. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2281–2284. ACM, 2011. pages 74

[10] M. Basu and T. K. Ho. *Data complexity in pattern recognition*. Springer Science & Business Media, 2006. pages 2, 9, 10, 12, 16, 81

[11] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on neural networks*, 5(4):537–550, 1994. pages 109

[12] R. Bekkerman, M. Bilenko, and J. Langford. *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011. pages 3, 49

[13] R. Bellman, R. Corporation, and K. M. R. Collection. *Dynamic Programming*. Princeton University Press, 1957. pages 4

[14] E. Bernadó-Mansilla and T. K. Ho. Domain of competence of xcs classifier system in complexity measurement space. *IEEE Transactions on Evolutionary Computation*, 9(1):82–104, 2005. pages 10

[15] D. Best and D. Roberts. Algorithm as 89: the upper tail probabilities of spearman's rho. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 24(3):377–379, 1975. pages 112

[16] V. Bolón-Canedo and A. Alonso-Betanzos. *Recent advances in ensembles for feature selection*, volume 147. Springer, 2018. pages 47

[17] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos. Feature selection and classification in multiple class datasets: An application to KDD Cup 99 dataset. *Expert Systems with Applications*, 38(5):5947–5957, 2011. pages 37

[18] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos. A distributed feature selection approach based on a complexity measure. In *International Work Conference on Artificial Neural Networks (in press)*, 2015. pages 74, 76, 78, 79

[19] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos. Recent advances and emerging challenges of feature selection in the context of big data. *Knowledge-Based Systems*, 2015. pages 3, 27, 81, 115

[20] V. Bolón-Canedo, N. Sánchez-Marono, A. Alonso-Betanzos, J. M. Benítez, and F. Herrera. A review of microarray datasets and applied feature selection methods. *Information Sciences*, 282:111–135, 2014. pages 10, 11

[21] V. Bolón-Canedo, N. Sánchez-Marono, and J. Cerviño-Rabuñal. Toward parallel feature selection from vertically partitioned data. In *ESANN*. Citeseer, 2014. pages 74, 76, 97

[22] A.-L. Boulesteix, R. Hable, S. Lauer, and M. J. Eugster. A statistical framework for hypothesis testing in real data comparison studies. *The American Statistician*, 69(3):201–212, 2015. pages 40

[23] U. Braga-Neto. 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*, 1(24):91–99, 2007. pages 23

[24] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. pages 154

[25] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984. pages 116

[26] G. Brown, A. Pocock, M.-J. Zhao, and M. Luján. Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. *Journal of machine learning research*, 13(Jan):27–66, 2012. pages 109, 117, 121

[27] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002. pages 24

[28] M. Chen, S. Mao, and Y. Liu. Big data: A survey. *Mobile networks and applications*, 19(2):171–209, 2014. pages 3

[29] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016. pages 59, 155

[30] C. Commons. Creative commons. urlhttps://search.creativecommons.org/, 2001. pages XXVII, 106

[31] K. Das, K. Bhaduri, and H. Kargupta. A local asynchronous distributed privacy preserving feature selection algorithm for large peer-to-peer networks. *Knowledge and information systems*, 24(3):341–367, 2010. pages 74

[32] M. Dash and H. Liu. Consistency-based search in feature selection. *Artificial intelligence*, 151(1-2):155–176, 2003. pages 152

[33] H. Daumé, J. M. Phillips, A. Saha, and S. Venkatasubramanian. Efficient protocols for distributed classification and optimization. In *International Conference on Algorithmic Learning Theory*, pages 154–168. Springer, 2012. pages 49

[34] R. C. de Amorim and C. Hennig. Recovering the number of clusters in data sets with noise features using feature rescaling factors. *Information Sciences*, 324:126–145, 2015. pages 125

[35] M. C. de Souto, A. C. Lorena, N. Spolaôr, and I. G. Costa. Complexity measures of supervised classifications tasks: a case study for cancer gene expression data. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2010. pages 14

[36] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008. pages 3, 47

[37] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(Jan):1–30, 2006. pages 40, 121

[38] T. Dick, M. Li, V. K. Pillutla, C. White, M. Balcan, and A. J. Smola. Data driven resource allocation for distributed learning. *CoRR*, abs/1512.04848, 2015. pages 48

[39] T. G. Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000. pages 49

[40] C. Ding and H. Peng. Minimum redundancy feature selection from microarray gene expression data. *Journal of bioinformatics and computational biology*, 3(02):185–205, 2005. pages 11

[41] E. Dougherty. Small sample issues for microarray-based classification. *Comparative and Functional Genomics*, 2(1):28–34, 2001. pages 23

[42] M. C. Du Plessis and M. Sugiyama. Semi-supervised learning of class balance under class-prior change by distribution matching. *Neural Networks*, 50:110–119, 2014. pages 4, 46

[43] C. Eiras-Franco, V. Bolón-Canedo, S. Ramos, J. González-Domínguez, A. Alonso-Betanzos, and J. Tourino. Multithreaded and spark parallelization of feature selection filters. *Journal of Computational Science*, 17:609–619, 2016. pages 74

[44] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936. pages 155

[45] G. Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research*, 3(Mar):1289–1305, 2003. pages 37

[46] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2011. pages 23

[47] M. Galar, A. Fernández, E. Barrenechea, and F. Herrera. Eusboost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. *Pattern Recognition*, 46(12):3460–3471, 2013. pages 25

[48] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science*, 286(5439):531–537, 1999. pages 11, 22

[49] B. Guo and M. S. Nixon. Gait feature subset selection by mutual information. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 39(1):36–46, 2009. pages 109

[50] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan. Deep learning with limited numerical precision. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1737–1746, 2015. pages 107

[51] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003. pages 99

[52] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror. Nips 2003 workshop on feature extraction. [Online; accessed May 2016]. http://clopinet.com/isabelle/Projects/NIPS2003/. pages 99, 116

[53] I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh. *Feature extraction: foundations and applications*, volume 207. Springer, 2006. pages 22

[54] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009. pages 28, 81

[55] M. A. Hall. Correlation-based feature selection for machine learning. 1999. pages 151

[56] M. A. Hall and L. A. Smith. Practical feature subset selection for machine learning. 1998. pages 152

[57] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally. Eie: efficient inference engine on compressed deep neural network. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 243–254. IEEE, 2016. pages 107

[58] A. d. Haro García et al. *Scaling data mining algorithms. Application to instance and feature selection*. Granada: Universidad de Granada, 2012. pages 76

[59] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, (9):1263–1284, 2008. pages 23

[60] T. K. Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (3):289–300, 2002. pages 18, 176

[61] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. J. Smola. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608, 2007. pages 57

[62] S. H. Huang. Supervised feature selection: A tutorial. *Artif. Intell. Research*, 4(2):22–37, 2015. pages 106

[63] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017. pages 107

[64] B. Institute. Cancer program data sets. [Online]. Available http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi. pages 27

[65] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018. pages 107

[66] A. Jain and D. Zongker. Feature selection: Evaluation, application, and small sample performance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(2):153–158, 1997. pages 22

[67] G. H. John, R. Kohavi, K. Pfleger, et al. Irrelevant features and the subset selection problem. In *Machine learning: proceedings of the eleventh international conference*, pages 121–129, 1994. pages 116

[68] K. Kira and L. A. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *AAAI*, volume 2, pages 129–134, 1992. pages 152

[69] J. Kittler, M. Hatef, R. P. Duin, and J. Matas. On combining classifiers. *IEEE transactions on pattern analysis and machine intelligence*, 20(3):226–239, 1998. pages 47, 51, 53, 55, 56, 58, 65

[70] I. Kononenko. Estimating attributes: analysis and extensions of relief. In *Machine Learning: ECML-94*, pages 171–182. Springer, 1994. pages 152

[71] P. Koopman. Design constraints on embedded real time control systems. 1990. pages 106

[72] D. D. Lewis. Feature selection and feature extraction for text categorization. In *Proceedings of the workshop on Speech and Natural Language*, pages 212–217. Association for Computational Linguistics, 1992. pages 109, 152

[73] M. Li, P. Vitányi, et al. *An introduction to Kolmogorov complexity and its applications*, volume 3. Springer, 2008. pages 2

[74] A. Liaw and M. Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002. pages 59

[75] M. Lichman. UCI machine learning repository, 2013. [Online; accessed April-2017]. pages 59, 115

[76] V. López, A. Fernández, S. García, V. Palade, and F. Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information sciences*, 250:113–141, 2013. pages 23

[77] A. C. Lorena, I. G. Costa, N. Spolaôr, and M. C. De Souto. Analysis of complexity indices for classification problems: Cancer gene expression data. *Neurocomputing*, 75(1):33–42, 2012. pages 11

[78] J. Luengo and F. Herrera. An automatic extraction method of the domains of competence for learning classifiers using data complexity measures. *Knowledge and Information Systems*, 42(1):147–180, 2015. pages 11

[79] L. Lusa et al. Class prediction for high-dimensional class-imbalanced data. *BMC bioinformatics*, 11(1):523, 2010. pages 23

[80] L. Lusa et al. Evaluation of smote for high-dimensional class-imbalanced microarray data. In *2012 11th International Conference on Machine Learning and Applications*, volume 2, pages 89–94. IEEE, 2012. pages 25

[81] N. Macià, E. Bernadó-Mansilla, A. Orriols-Puig, and T. K. Ho. Learner excellence biased by data set selection: A case for data characterisation and artificial data sets. *Pattern Recognition*, 46(3):1054–1066, 2013. pages 10

[82] S. Maldonado, R. Weber, and F. Famili. Feature selection for high-dimensional class-imbalanced data sets using support vector machines. *Information Sciences*, 286:228–246, 2014. pages 25

[83] S. McConnell and D. B. Skillicorn. Building predictors from vertically distributed data. In *Proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative research*, pages 150–162. IBM Press, 2004. pages 74

[84] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2015. R package version 1.6-7. pages 59

[85] R. A. Mollineda, J. S. Sánchez, and J. M. Sotoca. Data characterization for effective prototype selection. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 27–34. Springer, 2005. pages 10

[86] L. Morán-Fernández, V. Bolón-Canedo, and A. Alonso-Betanzos. A time efficient approach for distributed feature selection partitioning by features. In *Advances in Artificial Intelligence*, pages 245–254. Springer, 2015. pages 74

[87] J. G. Moreno-Torres, T. Raeder, R. Alaiz-RodríGuez, N. V. Chawla, and F. Herrera. A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530, 2012. pages 25

[88] M. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain. Machine learning at the network edge: A survey. *arXiv preprint arXiv:1908.00080*, 2019. pages 107

[89] L. Nanni and A. Lumini. Prototype reduction techniques: A comparison among different approaches. *Expert Systems with Applications*, 38(9):11820–11828, 2011. pages 134

[90] F. F. G. Navarro. *Feature selection in cancer research: microarray gene expression and in vivo 1h-mrs domains.* PhD thesis, Universitat Politècnica de Catalunya, 2011. pages 40

[91] V. Nikulin. On a solution for the high-dimensionality-small-sample-size regression problem with several different microarrays. *International journal of data mining and bioinformatics*, 9(3):221–234, 2014. pages 23

[92] O. Okun and H. Priisalu. Dataset complexity in gene expression based cancer classification using ensembles of k-nearest neighbors. *Artificial intelligence in medicine*, 45(2-3):151–162, 2009. pages 11

[93] A. Orriols-Puig, N. Macia, and T. K. Ho. Documentation for the data complexity library in c++. *Universitat Ramon Llull, La Salle*, 196:1–40, 2010. pages 12

[94] A. Orriols-Puig, N. Macia, and T. K. Ho. Documentation for the data complexity library in c++. *Universitat Ramon Llull, La Salle*, 196:1–40, 2010. pages 16

[95] L. Paninski. Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253, 2003. pages 108

[96] H. Peng, F. Long, and C. Ding. 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. pages 109, 153

[97] D. Peralta, S. del Río, S. Ramírez-Gallego, I. Triguero, J. M. Benitez, and F. Herrera. Evolutionary feature selection for big data classification: A mapreduce approach. *Mathematical Problems in Engineering*, 501:246139, 2015. pages 74

[98] D. Peteiro-Barral and B. Guijarro-Berdiñas. A survey of methods for distributed machine learning. *Progress in Artificial Intelligence*, 2(1):1–11, 2013. pages 45, 55

[99] G. Piatetsky-Shapiro and P. Tamayo. Microarray data mining: facing the challenges. *ACM SIGKDD Explorations Newsletter*, 5(2):1–5, 2003. pages 9, 21

[100] J. R. Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014. pages 154

[101] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009. pages 46

[102] S. Ramírez-Gallego, H. Mouriño-Talín, D. Martínez-Rego, V. Bolón-Canedo, J. M. Benítez, A. Alonso-Betanzos, and F. Herrera. An information theory-based feature selection framework for big data under apache spark. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(9):1441–1453, 2017. pages 74

[103] S. Ray, J. Park, and S. Bhunia. Wearables, implants, and internet of things: the technology needs in the evolving landscape. *IEEE Transactions on Multi-Scale Computing Systems*, 2(2):123–128, 2016. pages 105

[104] I. Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001. pages 154

[105] M. Á. Rodríguez, A. Fernández, A. Peregrín, and F. Herrera. A review of distributed data models for learning. *Hybrid Artificial Intelligent Systems*, page 88, 2017. pages 48

[106] Y. Saeys, I. Inza, and P. Larrañaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007. pages 10, 22, 73

[107] J. S. Sánchez, R. A. Mollineda, and J. M. Sotoca. An analysis of how training data complexity affects the nearest neighbor classifiers. *Pattern Analysis and Applications*, 10(3):189–201, 2007. pages 10

[108] K. Sechidis, L. Azzimonti, A. Pocock, G. Corani, J. Weatherall, and G. Brown. Efficient feature selection using shrinkage estimators. *Machine Learning*, 108(8):1261–1286, Sep 2019. pages 109

[109] K. Sechidis and G. Brown. Simple strategies for semi-supervised feature selection. *Machine Learning*, 107(2):357–395, 2018. pages 113

[110] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016. pages 5

[111] D. B. Skillicorn and S. M. McConnell. Distributed prediction from vertically partitioned data. *Journal of Parallel and Distributed computing*, 68(1):16–36, 2008. pages 74

[112] F. W. Smith. Pattern classifier design by linear programming. *IEEE transactions on computers*, 100(4):367–372, 1968. pages 16

[113] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437, 2009. pages 157

[114] R. J. Solomonoff. The kolmogorov lecture the universal distribution and machine learning. *The Computer Journal*, 46(6):598–601, 2003. pages 2

[115] A. Statnikov, I. Tsamardinos, Y. Dosbayev, and C. F. Aliferis. Gems: a system for automated cancer diagnosis and biomarker discovery from microarray gene expression data. *International journal of medical informatics*, 74(7-8):491–503, 2005. pages 27, 80

[116] G. J. Székely and M. L. Rizzo. Testing for equal distributions in high dimension. *InterStat*, 5:1–6, 2004. pages 52

[117] G. J. Székely and M. L. Rizzo. Energy statistics: A class of statistics based on distances. *Journal of statistical planning and inference*, 143(8):1249–1272, 2013. pages 52

[118] M. Tesmer and P. A. Estévez. Amifs: Adaptive feature selection by using mutual information. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 1, pages 303–308. IEEE, 2004. pages 109

[119] S. Tschiatschek and F. Pernkopf. Parameter learning of bayesian network classifiers under computational constraints. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 86–101. Springer, 2015. pages 107, 110, 125, 127

[120] G. Tsoumakas and I. Vlahavas. Distributed data mining. *Encyclopedia of Data Warehousing and Mining*, 2009. pages 49

[121] A. Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106(2):58, 2004. pages 4

[122] A. S. University. Feature selection datasets. [Online]. Available: http://featureselection.asu.edu/datasets.php. pages 27, 80

[123] V. Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013. pages 154

[124] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. ISBN 0-387-95457-0. pages 59

[125] Y. Wang, I. V. Tetko, M. A. Hall, E. Frank, A. Facius, K. F. Mayer, and H. W. Mewes. Gene selection from microarray data for cancer classification—a machine learning approach. *Computational biology and chemistry*, 29(1):37–46, 2005. pages 11

[126] D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996. pages 2

[127] E. P. Xing, M. I. Jordan, R. M. Karp, et al. Feature selection for high-dimensional genomic microarray data. In *ICML*, volume 1, pages 601–608. Citeseer, 2001. pages 11

[128] H. Yang and G. Churchill. Estimating p-values in small microarray experiments. *Bioinformatics*, 23(1):38–43, 2007. pages 23

[129] H. H. Yang and J. Moody. Data visualization and feature selection: New algorithms for nongaussian data. In *Advances in neural information processing systems*, pages 687–693, 2000. pages 109, 153

[130] Z. Zhao and H. Liu. Searching for interacting features. In *IJCAI*, volume 7, pages 1156–1161, 2007. pages 152

Appendix A

Methods

This appendix describes the feature selection methods and classification algorithms used in this thesis.

A.1. Feature selection methods

Feature selection methods have traditionally been categorized as filter, wrappers and embedded methods. Filter methods are based on performance evaluation metric calculated directly from the data, without direct feedback from predictors that will finally be used on data with reduced number of features. As mentioned in Chapter 4, we have used these algorithms since they are usually computationally less expensive than wrappers or embedded methods. In this section, the filters used throughout this thesis are described.

- **Correlation-based Feature Selection** (CFS). This simple multivariate filtering algorithm ranks feature subsets according to a correlation based heuristic evaluation function [55]. The evaluation function is biased towards subsets containing features that are highly correlated with the class and uncorrelated with each other. Irrelevant features with low correlation with the class are ignored. Redundant features are screened out as they would be highly correlated with one or more of the remaining features. The acceptance of a feature depends on the extent to which it predicts classes in areas of the instance space not already predicted by other features.

- **Consistency-based filter** (CONS) [32]. This filter evaluates the worth of a features subset according to consistency in class values when training samples are projected onto the features subset. The algorithm generates a random subset S from the number of features encountered in each round. If there are fewer features in S than in the current best set, the data with the features prescribed in S are checked against the inconsistency criterion and, if the inconsistency rate is below a pre-specified threshold, S becomes the new current best set.

- **INTERACT** (INT) [130]. This algorithm is based on symmetrical uncertainty (SU) and it also includes the consistency contribution. It consists of two main 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.

- **Information gain** (IG) [56]. This filter evaluates the features according to their information gain and considers a single feature at a time. It provides an orderly classification of all features, and then a threshold is required to select a certain number of them according to the order obtained.

- **ReliefF** [70]. This algorithm, an extension of the original Relief [68], adds the ability of dealing with noisy, incomplete and multiclass datasets. The key idea of this algorithm is to estimate features according to how well their values distinguish among the instances that are near to each other. This method may be applied in all situations, has low bias, includes iteration among features and may capture local dependencies which other methods miss.

A.1.1. Information theory-based feature selection methods

- **Mutual Information Maximisation** (MIM) [72]. This method ranks the features by their MI score, and selects the top k features, where k is decided by some predefined need for a certain number of features or some other stopping criterion.

$$MIM(X_k) = I(X_k; Y) \tag{A.1}$$

An important limitation is that this assumes that each feature is independent of all other features and effectively ranks the features in descending order of their MI content. Thus, this approach does not take into account the redundancy between the features.

- **Joint Mutual Information** (JMI) [129]. This method is focused on increasing complementary information between features. The JMI score for feature X_k is:

$$JMI(X_k) = \sum_{X_j \in S} I(X_k X_j; Y) \tag{A.2}$$

This is the information between the targets and a joint random variable $X_k X_j$, defined by pairing the candidate X_k with each feature previously selected. The idea is if the candidate feature is "complementary" with existing features, we should include it.

- **minimum Redundancy Maximum Relevance** (mRMR) [96]. The multivariate filter selects features that have the highest relevance with the target class and are also minimally redundant, i.e. it selects features that are maximally dissimilar to each other. Both optimization criteria (maximum-relevance and minimum-redundancy) are based on mutual information. Let S denote the subset of features we are seeking:

$$mRMR(X_k) = I(X_k; Y) - \sum_{X_j \in S} I(X_k X_j; Y) \tag{A.3}$$

The mRMR criterion, like JMI, has a strong belief in the pairwise independence assumptions as the feature set S grows.

A.2. Classification algorithms

This section describes the most common classification algorithms which are used along this thesis.

- **Naive Bayes** (NB) [104]. This simple probabilistic classifier is 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.

- **Support Vector Machine** (SVM) [123]. This learning algorithm, used for classification, regression and other tasks, was proposed by Vapnik. More formally, a SVM constructs a hyperplane or set of hyperplanes in a high —or finite— dimensional space. Intuitively, good separation is achieved by the hyperplane with the greatest distance to the nearest training data point of any class (functional margin), since in general, the larger the margin, the lower the generalization error of the classifier.

- **C4.5** [100]. This classifier was developed by Quinlan as an extension of the ID3 algorithm (both are based on decision tree concepts). A decision tree classifies a pattern by means of a descending filtering until is found a leaf, that points to the corresponding classification. One of the improvements with respect to ID3 is that C4.5 can deal with both numerical and symbolic data.

- ***k*-Nearest Neighbor** (k-NN) [3]. This classification strategy is an example of a "lazy learner". An object is classified by majority vote of its neighbors and is assigned to the most common class among its k nearest neighbors. If $k = 1$, 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.

- **Random Forest** (RF) [24]. Random forests are a combination of random trees, which are decision trees generated in a specific way to obtain diversity among the trees. Each random tree is trained on a bootstrap replicate, i.e., a example obtained from n training samples by randomly selecting n samples with replacement. This procedure is referred to as "bagging". Besides, only a randomly selected subset of the available features is considered when choosing each interior split. The samples that were missing in the bootstrap replicate, for a specific tree, are said to be out-of-bag for that tree.

- **Fisher's linear discriminant** (LDA) [44]. Fisher's linear discriminant is perhaps the oldest classification method in the literature and yet is still being used in modern applications. LDA was originally suggested by Fisher where the basic idea was to find an optimal linear projection of the data in which the optimality measure is the ratio of the between-class variance to the within-class variance on the projected data.

- **eXtreme Gradient Boosting** (XGB) [29]. XGB is a boosting tree algorithm that is an enhancement over tree bagging methodologies. The basic philosophy of bagging is based on combining three concepts: (i) creation of multiple datasets; (ii) building of multiple trees and (iii) bootstrap aggregation or bagging. It adopts a divide-and-conquer approach to capture non-linearities in the data and perform pattern recognition. Its core principle is that a group of "weak learner" combined, can form a "strong predictor" model.

- **Multinomial logistic regression** (Mult). This algorithm is an extension of the logistic regression for multiclass classification tasks.

Appendix B

Supplementary material

This appendix reports the experimental results achieved in this thesis.

Regarding Chapter 3, the same numerical analysis performed in Section 3.4.2 with the accuracy values has been carried out for recall and precision values, two alternative performance measures. The attained results are shown in this Appendix B. In the case of binary classification, recall measures the effectiveness of a classifier to identify correctly classified positive instances (sensitivity), while precision evaluates the class agreement of the instance labels with the positive labels given by the classifier. One possible way to extend these concepts to the multi-class classification task is to obtain the averages of these measures calculated over all the classes $\{C_1, \dots, C_m\}$. This generalization approach is known by *macro-averaging* [113]. This way, we have

$$\begin{aligned} Precision &= \frac{1}{m}\sum_{i=1}^{m}\frac{tp_i}{tp_i + fp_i}, \\ Recall &= \frac{1}{m}\sum_{i=1}^{m}\frac{tp_i}{tp_i + fn_i}, \end{aligned}$$

with m the number of classes in the dataset, and tp_i denoting the number of true positive for C_i, and fp_i and fn_i the false positive and false negative counts, respectively.

The results attained for these alternative criteria are displayed below, using the same scheme of tables and figures as in Section 3.4.2 for accuracy. It can be seen that very similar results are also obtained, thus supporting the main conclusions of our work.

Table B.1: Average recall values conditional on classifier type.

	Classifier				
Model	RF	SVM	XGB	LDA	Mult
BALANCED					
pNW	0.7154	0.5912	0.6991	0.5788	0.6128
pIW	0.7240	0.5968	0.7079	0.5846	0.6211
UW	0.7197	0.5935	0.7031	0.5788	0.6109
UNBALANCED					
pNW	0.7029	0.5890	0.6926	0.5777	0.6093
pIW	0.7154	0.6008	0.7063	0.5915	0.6247
UW	0.7094	0.5869	0.6998	0.5797	0.6078
MEAN					
pNW	0.7091	0.5901	0.6959	0.5783	0.6110
pIW	0.7197	0.5988	0.7071	0.5880	0.6229
UW	0.7146	0.5902	0.7015	0.5792	0.6094
ND	0.7676	0.6794	0.7477	0.6418	0.6515

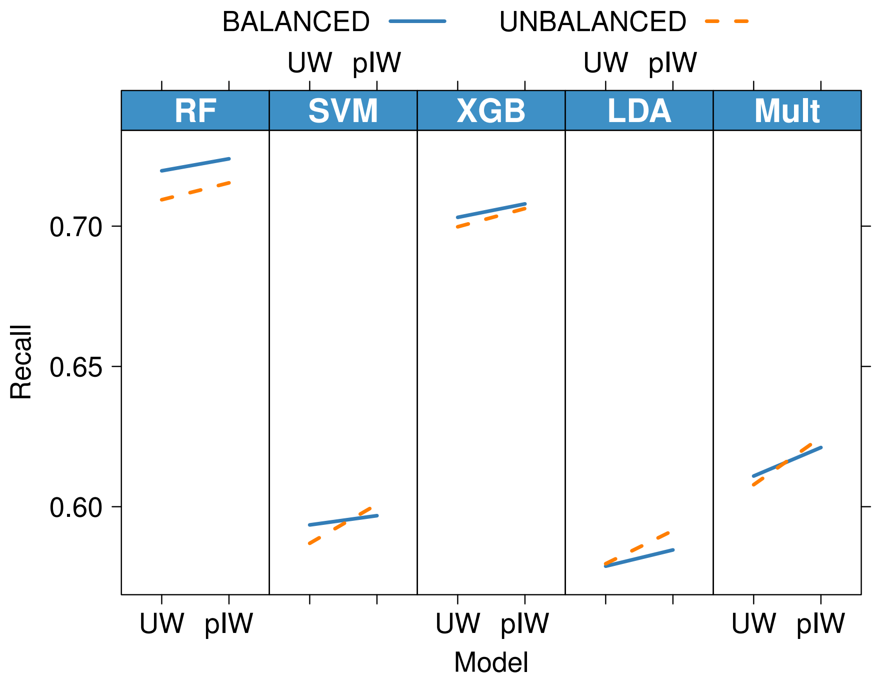


Figure B.1: Recall-based interaction plot to check the joint effect of classifier, learning model and scenario.

Table B.2: Average recall values conditional on the decision rules.

	Decision rule					
Model	MAJ	MAX	MIN	PROD	SUM	Mean
BALANCED						
pNW	0.6505	0.6291	0.6237	0.6401	0.6539	0.6395
pIW	0.6633	0.6506	0.6119	0.6399	0.6687	0.6469
UW	0.6507	0.6323	0.6262	0.6402	0.6567	0.6412
Mean	0.6548	0.6374	0.6206	0.6401	0.6598	0.6425
UNBALANCED						
pNW	0.6493	0.6245	0.6060	0.6353	0.6564	0.6343
pIW	0.6733	0.6582	0.5895	0.6351	0.6826	0.6478
UW	0.6527	0.6257	0.6086	0.6353	0.6613	0.6367
Mean	0.6584	0.6362	0.6014	0.6353	0.6668	0.6396
ND	—	—	—	—	—	0.6976

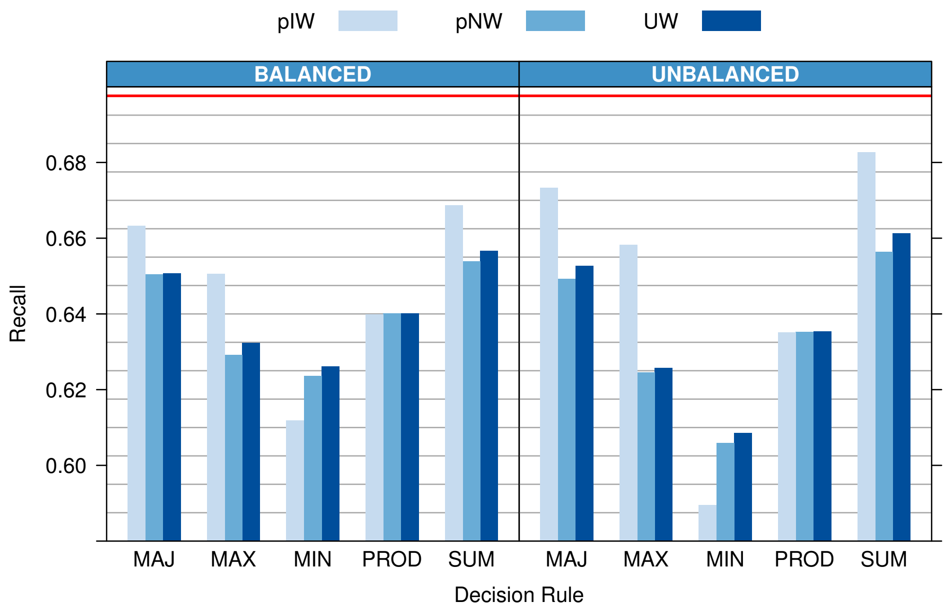


Figure B.2: Average recall values aggregated by decision rules. The horizontal red line indicates the average recall for the ND model.

Table B.3: Average precision values conditional on classifier type.

	Classifier				
Model	RF	SVM	XGB	LDA	Mult
BALANCED					
pNW	0.7016	0.5631	0.6896	0.5833	0.6118
pIW	0.7113	0.5723	0.6989	0.5895	0.6200
UW	0.7060	0.5665	0.6935	0.5818	0.6103
UNBALANCED					
pNW	0.6875	0.5577	0.6827	0.5815	0.6098
pIW	0.7001	0.5760	0.6956	0.5948	0.6249
UW	0.6929	0.5569	0.6879	0.5819	0.6090
MEAN					
pNW	0.6945	0.5604	0.6862	0.5824	0.6108
pIW	0.7057	0.5741	0.6973	0.5922	0.6224
UW	0.6994	0.5617	0.6907	0.5819	0.6097
ND	0.7516	0.6586	0.7392	0.6405	0.6496

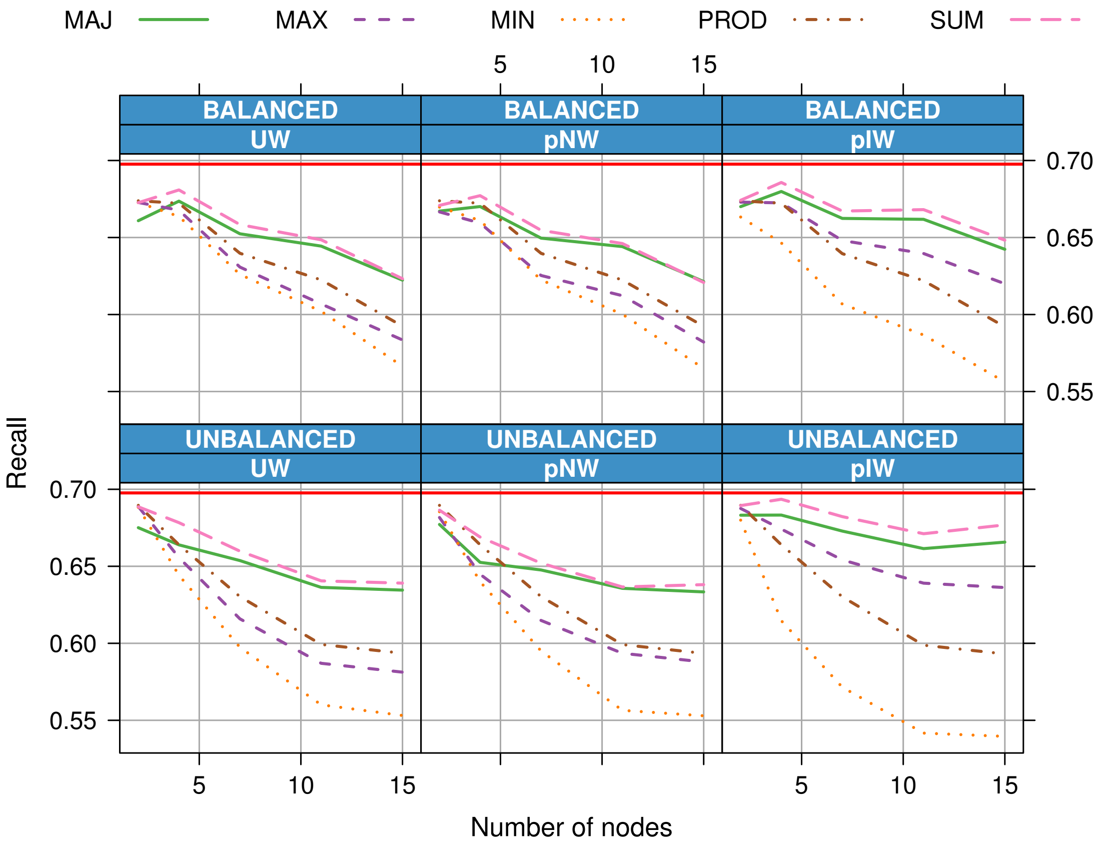


Figure B.3: Average recall as function of the partition size. The horizontal red lines indicate the average recall for the ND model.

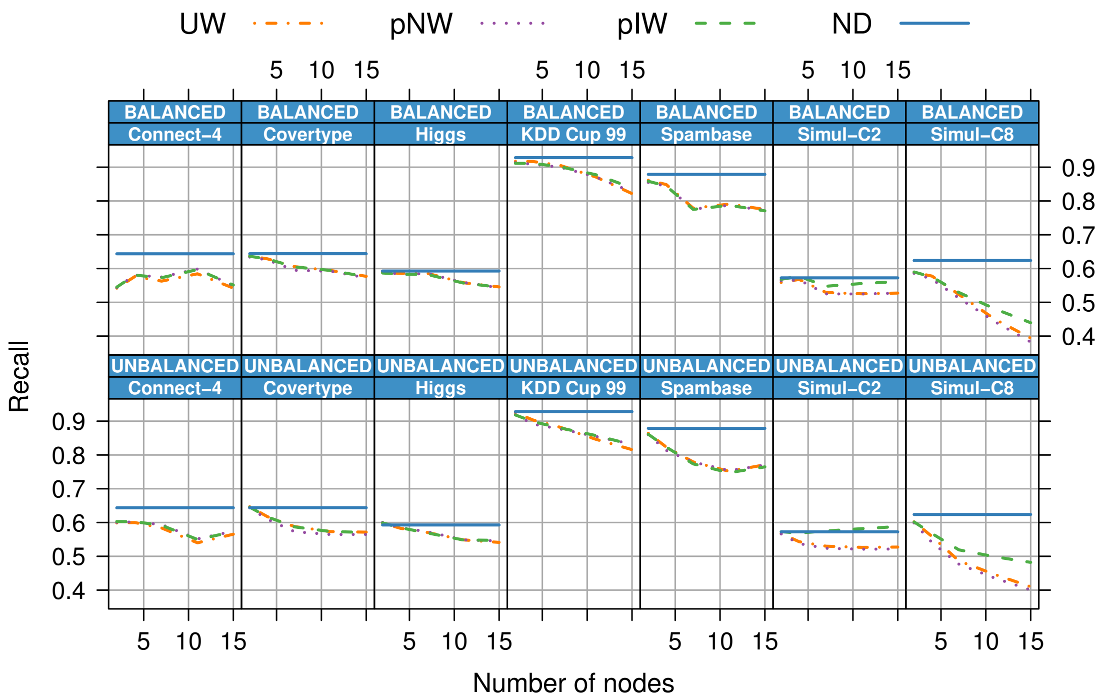


Figure B.4: Average recall as function of the partition size for each data set. The horizontal blue lines indicate the average recall for the ND model.

Table B.4: Average precision values conditional on the decision rules.

	Decision rule					
Model	MAJ	MAX	MIN	PROD	SUM	Mean
BALANCED						
pNW	0.6392	0.6239	0.6145	0.6282	0.6436	0.6299
pIW	0.6550	0.6459	0.6030	0.6280	0.6601	0.6384
UW	0.6408	0.6266	0.6166	0.6283	0.6459	0.6316
Mean	0.6450	0.6321	0.6113	0.6282	0.6499	0.6333
UNBALANCED						
pNW	0.6378	0.6193	0.5945	0.6226	0.6451	0.6238
pIW	0.6637	0.6531	0.5799	0.6224	0.6724	0.6383
UW	0.6420	0.6178	0.5985	0.6226	0.6477	0.6257
Mean	0.6478	0.6300	0.5910	0.6225	0.6551	0.6293
ND	—	—	—	—	—	0.6879

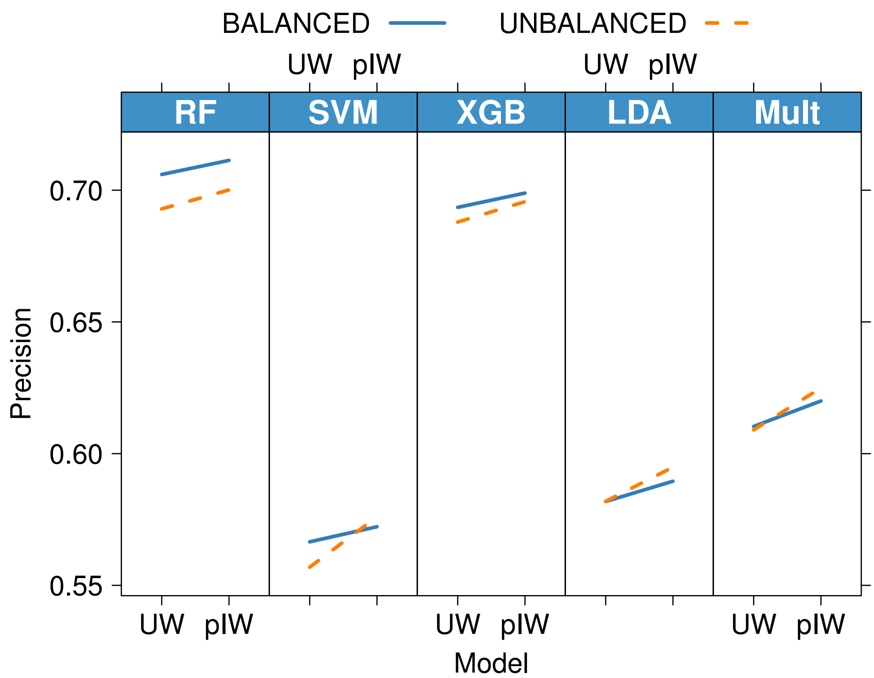


Figure B.5: Precision-based interaction plot to check the joint effect of classifier, learning model and scenario.

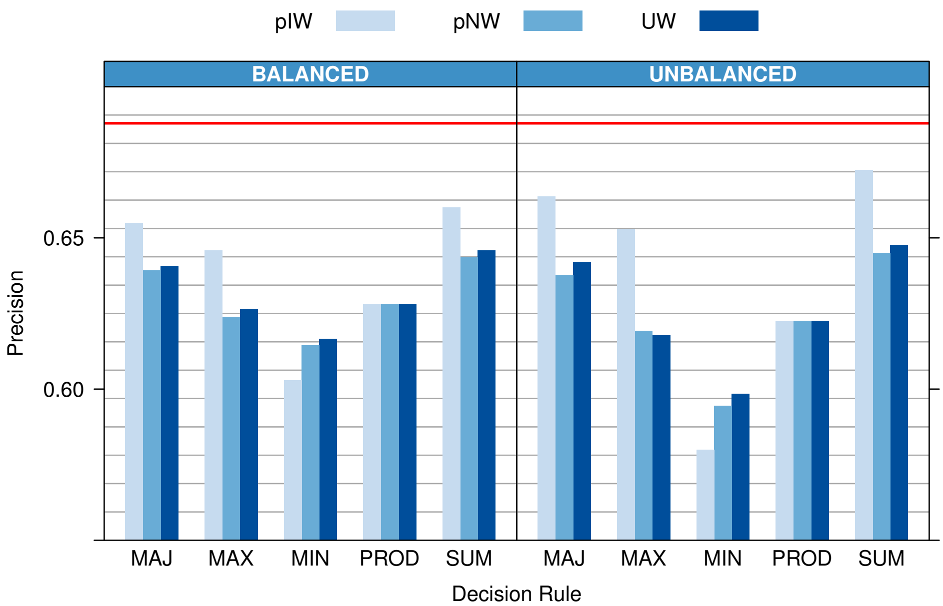


Figure B.6: Average precision values aggregated by decision rules. The horizontal red line indicates the average precision for the ND model.

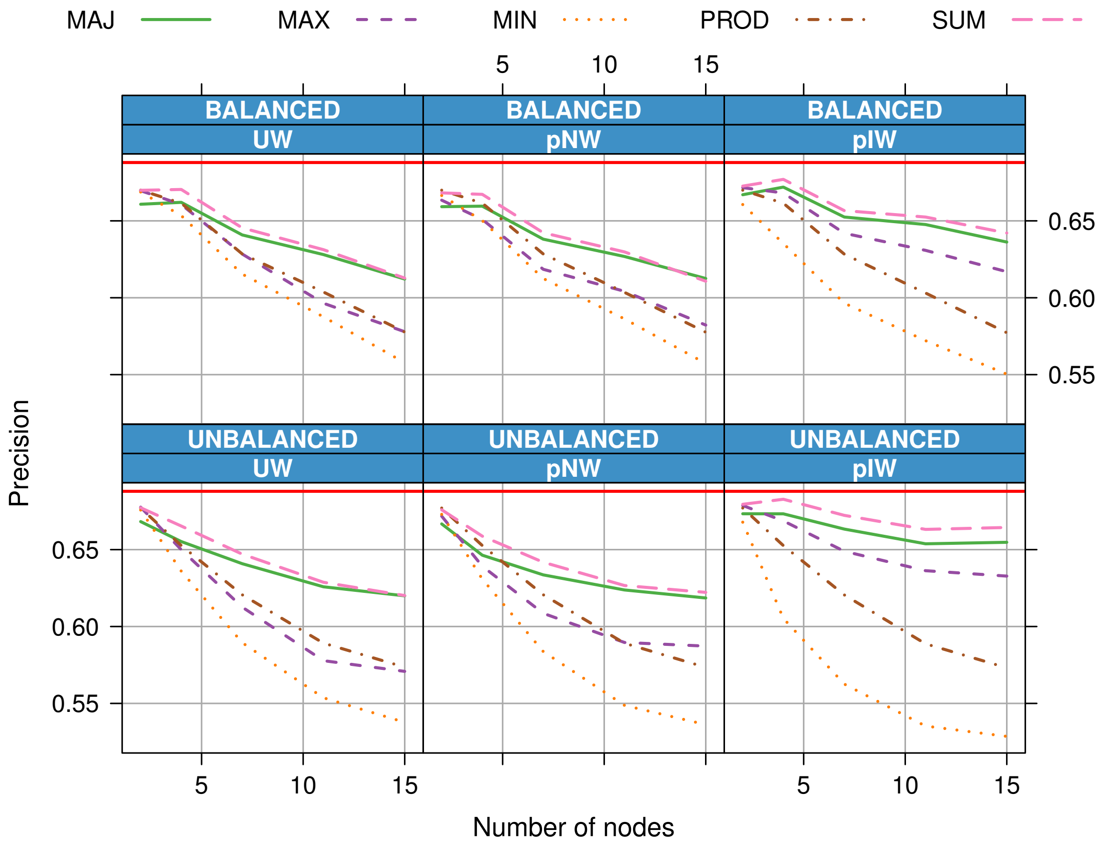


Figure B.7: Average precision as function of the partition size. The horizontal red lines indicate the average precision for the ND model.

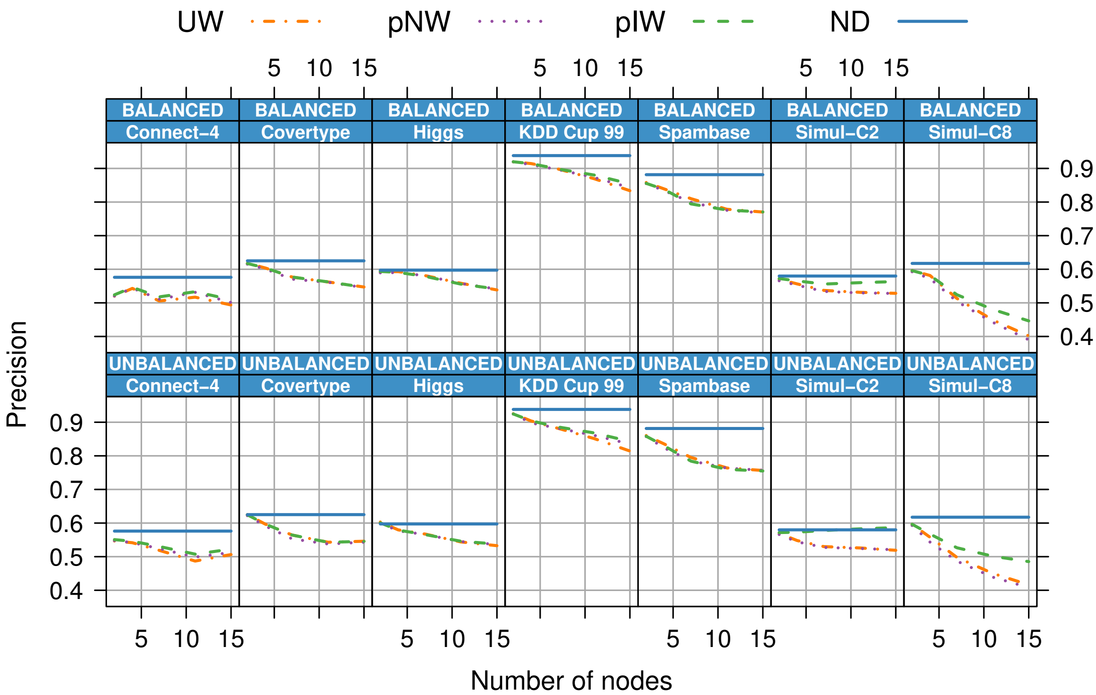


Figure B.8: Average precision as function of the partition size for each data set. The horizontal blue lines indicate the average precision for the ND model.

In relation to Chapter 4, Tables B.5, B.6, B.7 and B.8 depict classification accuracy for horizontal and vertical partitioning, respectively. Tables B.9 and B.10 show classification accuracy rates achieved by the vertical distributed approach D-F1 including a certain level of overlap in the packets.

Table B.5: Test classification accuracy achieved by the centralized and distributed approaches of horizontal partitioning using C4.5 and NB algorithms, $\alpha = 0.75$.

			Connect4	Isolet	Madelon	Ozone	Spambase	Mnist
C4.5	CFS	C	61.22	81.59	80.50	97.63	81.16	86.99
		D-F1	61.25	81.53	80.62	96.09	79.73	88.55
		D-F2	61.72	56.45	80.63	96.45	79.73	88.80
		D-N2	62.36	81.85	79.63	97.16	79.73	88.80
	INT	C	60.48	78.96	80.63	96.92	78.16	87.24
		D-F1	61.25	79.35	80.62	96.92	80.44	88.45
		D-F2	61.66	79.35	80.05	97.75	80.05	88.36
		D-N2	62.72	78.77	80.63	96.69	80.05	88.36
	Cons	C	60.49	56.00	80.63	98.70	84.62	87.00
		D-F1	61.25	72.87	80.62	97.75	85.27	89.46
		D-F2	61.66	66.71	80.63	98.70	80.83	89.06
		D-N2	62.72	65.43	80.63	98.70	80.83	89.06
	IG	C	63.90	81.40	72.75	98.22	83.83	87.83
		D-F1	62.27	79.41	80.62	97.87	81.68	87.77
		D-F2	62.27	76.39	80.63	97.75	85.40	87.77
		D-N2	62.09	77.29	79.63	97.75	85.40	87.77
	ReliefF	C	63.49	79.54	73.88	98.11	78.81	87.34
		D-F1	63.00	81.53	84.12	95.98	84.88	88.06
		D-F2	63.00	65.30	84.13	98.60	84.22	88.09
		D-N2	60.26	80.50	84.13	98.46	84.22	88.09
NB	CFS	C	60.28	75.05	71.75	78.22	57.69	71.88
		D-F1	58.83	75.30	70.50	77.63	58.87	73.49
		D-F2	60.56	45.35	70.50	72.54	58.87	73.70
		D-N2	56.73	74.66	70.63	72.90	58.87	73.70
	INT	C	53.85	71.26	70.00	78.22	57.95	70.94
		D-F1	58.83	69.60	70.00	76.21	78.42	71.30
		D-F2	59.16	69.98	70.00	80.71	61.73	71.35
		D-N2	55.85	69.85	70.00	75.27	61.73	71.35
	Cons	C	54.12	42.78	70.00	98.70	91.00	72.78
		D-F1	58.83	64.91	70.00	98.46	92.05	74.61
		D-F2	59.16	52.73	70.00	98.70	91.00	73.33
		D-N2	55.85	48.49	70.00	98.70	91.00	73.33
	IG	C	60.42	69.34	70.38	74.08	76.53	70.74
		D-F1	60.20	66.77	70.50	78.46	66.95	68.07
		D-F2	60.20	60.62	70.50	77.99	90.35	68.07
		D-N2	57.27	66.45	70.63	74.67	90.35	68.07
	ReliefF	C	60.42	62.67	68.63	71.36	41.85	69.82
		D-F1	60.50	53.69	72.25	66.86	92.05	70.89
		D-F2	60.50	33.42	72.25	59.41	92.51	70.86
		D-N2	59.11	53.18	72.25	57.63	92.50	70.86

Table B.6: Test classification accuracy achieved by the centralized and distributed approaches of horizontal partitioning using kNN and SVM algorithms, $\alpha = 0.75$.

			Connect4	Isolet	Madelon	Ozone	Spambase	Mnist
kNN	CFS	C	53.90	56.00	85.63	96.45	79.14	87.93
		D-F1	53.68	54.65	88.50	94.56	79.92	91.57
		D-F2	52.76	52.21	88.50	94.79	79.92	91.87
		D-N2	59.67	57.92	86.13	94.20	79.92	91.87
	INT	C	58.27	52.92	88.75	94.44	79.73	86.87
		D-F1	53.68	50.42	88.75	94.79	76.92	91.72
		D-F2	57.61	53.18	88.75	98.70	76.79	93.74
		D-N2	60.77	53.43	88.75	93.37	76.34	91.71
	Cons	C	58.06	49.90	88.75	98.70	80.83	87.36
		D-F1	53.68	57.41	88.75	95.50	76.79	95.14
		D-F2	57.61	65.81	88.75	98.70	76.79	93.75
		D-N2	60.77	62.73	88.75	91.60	76.79	93.74
	IG	C	51.29	54.78	74.25	95.98	78.62	89.63
		D-F1	54.52	61.83	88.50	94.79	77.71	90.97
		D-F2	54.52	62.16	88.50	93.85	78.03	90.97
		D-N2	58.89	63.05	86.13	94.67	78.03	90.97
	ReliefF	C	61.81	59.14	75.25	95.98	76.99	89.97
		D-F1	57.01	55.93	88.38	96.33	80.18	91.77
		D-F2	57.01	50.42	88.38	95.62	82.72	91.19
		D-N2	56.67	55.23	88.38	96.09	82.72	91.19
SVM	CFS	C	60.42	83.54	66.50	98.70	85.85	79.58
		D-F1	60.42	82.30	66.75	98.70	85.46	81.49
		D-F2	60.42	42.85	66.75	98.70	85.46	81.81
		D-N2	60.42	82.75	67.13	98.70	85.46	81.81
	INT	C	60.42	73.83	66.38	98.70	80.31	78.54
		D-F1	60.42	75.18	66.38	98.70	81.10	80.87
		D-F2	60.42	75.50	66.38	98.70	81.88	81.01
		D-N2	60.42	74.54	66.38	98.70	81.88	81.01
	Cons	C	60.42	31.17	66.38	98.70	81.88	74.14
		D-F1	60.42	60.49	66.38	98.70	81.16	80.52
		D-F2	60.42	41.31	66.38	98.70	80.38	79.49
		D-N2	60.42	33.93	66.38	98.70	80.38	79.15
	IG	C	60.42	82.94	67.13	98.70	83.83	78.28
		D-F1	60.42	80.12	66.75	98.70	83.38	79.15
		D-F2	60.42	71.91	66.75	98.70	83.51	79.15
		D-N2	60.42	79.79	67.13	98.70	83.51	79.15
	ReliefF	C	60.42	84.61	67.50	98.70	81.94	75.43
		D-F1	60.42	81.98	67.25	98.70	83.77	75.86
		D-F2	60.42	66.90	67.25	98.70	85.59	75.67
		D-N2	60.42	80.56	67.25	98.70	85.60	75.67

Table B.7: Test classification accuracy achieved by the centralized and distributed approaches of vertical partitioning using C4.5 and NB algorithms, $\alpha = 0.75$.

			Isolet	Madelon	Mnist	Breast	Gli85	CLL-SUB-111	Lung cancer	11-Tumors
C4.5	CFS	C	80.61	73.34	85.36	68.42	75.86	59.46	77.94	53.45
		D-F1	83.90	79.63	87.07	73.68	75.86	67.57	83.82	60.34
		D-F2	82.68	72.75	86.85	73.68	75.86	67.57	83.82	67.24
		D-N2	83.51	80.50	86.89	73.68	75.86	62.16	83.82	65.52
	INT	C	76.76	77.94	84.77	78.95	75.86	59.46	77.94	58.62
		D-F1	79.99	74.38	89.14	73.68	75.86	72.97	75.00	67.24
		D-F2	73.51	78.50	89.93	73.68	75.86	56.76	79.41	60.34
		D-N2	80.12	80.63	89.41	73.86	75.86	67.57	85.29	65.52
	Cons	C	54.04	78.39	85.40	68.42	82.76	72.97	83.82	46.55
		D-F1	79.79	77.75	86.31	78.95	75.86	70.27	76.47	63.79
		D-F2	60.81	73.13	84.63	73.68	75.86	62.16	76.47	63.79
		D-N2	79.03	80.63	89.31	73.68	75.86	70.27	80.88	62.07
	IG	C	79.00	79.63	86.69	52.63	75.86	62.16	86.76	56.90
		D-F1	82.23	79.63	88.80	73.68	75.86	70.27	82.35	65.52
		D-F2	82.49	72.75	88.66	73.68	75.86	62.16	82.35	65.52
		D-N2	81.98	80.50	88.60	73.68	75.86	62.16	83.82	67.24
	ReliefF	C	79.02	82.93	85.59	73.68	75.86	72.97	88.24	60.34
		D-F1	84.41	69.50	89.70	52.63	75.86	72.97	82.35	65.52
		D-F2	83.32	69.38	88.17	57.89	75.86	62.16	82.35	63.79
		D-N2	84.73	84.38	87.76	57.89	75.86	62.16	85.29	67.24
NB	CFS	C	72.44	69.94	72.18	36.84	86.21	64.86	92.65	84.48
		D-F1	76.78	70.63	70.24	36.84	82.76	59.46	95.59	82.76
		D-F2	75.95	70.38	70.62	36.84	82.76	59.46	95.59	82.76
		D-N2	75.75	71.75	69.71	36.84	86.21	62.16	95.59	81.03
	INT	C	67.17	69.73	69.87	36.84	86.21	67.57	94.12	82.76
		D-F1	73.00	67.75	71.11	36.84	86.21	64.86	94.12	82.76
		D-F2	67.16	68.13	72.61	36.84	86.21	70.27	94.12	81.03
		D-N2	72.61	70.50	72.51	36.84	86.21	59.46	92.65	81.03
	Cons	C	40.19	69.81	71.78	36.84	82.76	70.27	85.29	58.62
		D-F1	68.51	67.25	73.04	36.84	86.21	67.58	91.18	86.21
		D-F2	44.52	68.00	73.50	36.84	86.21	62.16	92.65	84.48
		D-N2	67.48	70.50	74.21	36.84	79.31	64.86	94.12	84.48
	IG	C	69.62	69.78	68.80	36.84	79.31	70.27	88.23	86.21
		D-F1	70.24	70.63	69.61	36.84	82.76	64.86	91.18	82.76
		D-F2	71.52	70.38	69.35	36.84	82.76	62.16	91.18	82.76
		D-N2	70.05	71.75	71.06	36.84	86.21	67.57	91.18	84.48
	ReliefF	C	64.67	69.64	69.58	84.21	86.21	62.16	88.24	75.86
		D-F1	70.69	67.50	71.08	84.21	82.76	70.27	91.18	82.76
		D-F2	70.36	68.38	69.21	89.47	82.76	59.46	91.18	82.76
		D-N2	69.17	69.75	69.40	84.21	82.76	70.27	91.18	81.03

Table B.8: Test classification accuracy achieved by the centralized and distributed approaches of vertical partitioning using KNN and SVM algorithms, $\alpha = 0.75$.

			Isolet	Madelon	Mnist	Breast	Gli85	CLL-SUB-111	Lung cancer	11-Tumors
kNN	CFS	C	55.63	68.86	86.34	63.16	75.86	72.97	95.59	89.66
		D-F1	60.81	86.13	87.56	78.95	86.21	72.97	95.59	82.76
		D-F2	56.25	74.25	87.68	78.95	86.21	75.68	94.12	84.48
		D-N2	61.32	85.63	86.92	73.68	86.21	72.97	94.12	84.48
	INT	C	46.88	72.95	85.04	52.63	79.31	62.16	94.12	87.93
		D-F1	51.44	76.00	93.24	84.21	86.21	67.57	95.59	84.48
		D-F2	50.99	80.88	94.03	78.95	86.21	75.68	95.59	84.48
		D-N2	49.78	88.50	94.07	73.68	86.21	72.97	92.65	82.76
	Cons	C	53.05	75.04	85.42	47.37	82.76	75.68	83.82	55.17
		D-F1	63.44	71.25	89.09	73.68	86.21	72.97	94.12	87.93
		D-F2	57.86	76.00	87.17	63.16	86.21	67.57	92.65	86.21
		D-N2	61.00	88.50	95.58	78.95	86.21	67.57	94.12	82.76
	IG	C	53.94	78.07	89.35	68.42	89.66	62.16	95.59	82.76
		D-F1	55.36	86.13	92.64	78.95	86.21	75.68	97.06	84.48
		D-F2	56.45	74.25	92.62	78.95	86.21	72.97	97.06	82.76
		D-N2	55.10	85.63	92.43	78.95	86.21	70.27	95.59	84.48
	ReliefF	C	56.69	89.32	89.27	73.68	82.76	75.68	95.59	82.76
		D-F1	60.87	69.75	94.00	73.68	82.76	83.78	94.12	82.76
		D-F2	60.55	67.75	91.22	89.47	82.76	83.78	94.12	82.76
		D-N2	58.69	90.25	90.92	78.95	82.76	67.57	94.12	82.76
SVM	CFS	C	82.36	65.23	80.15	68.42	86.21	72.97	94.12	89.66
		D-F1	85.44	67.13	78.82	78.95	86.21	67.57	98.53	87.93
		D-F2	84.16	67.13	78.60	78.95	86.21	64.87	97.06	87.93
		D-N2	84.67	66.50	78.36	63.16	86.21	62.16	97.06	87.93
	INT	C	72.53	65.01	78.44	73.68	86.21	67.57	94.12	86.21
		D-F1	83.07	67.00	78.77	78.95	86.21	72.97	98.53	87.93
		D-F2	68.76	66.88	79.90	78.95	86.21	67.57	98.53	87.93
		D-N2	78.06	66.75	79.59	63.16	86.21	67.58	98.53	89.66
	Cons	C	29.29	65.04	74.42	31.59	89.66	62.16	79.41	48.28
		D-F1	69.47	67.00	76.07	78.95	86.21	70.27	97.06	86.21
		D-F2	33.80	67.00	76.03	78.95	86.21	70.27	94.12	87.93
		D-N2	69.53	66.75	78.69	78.95	82.76	67.57	94.12	89.66
	IG	C	81.09	65.90	78.60	78.95	86.21	64.85	95.59	84.48
		D-F1	85.12	67.13	79.85	78.95	86.21	72.97	97.06	86.21
		D-F2	85.76	67.13	80.07	78.95	86.21	70.27	97.06	84.48
		D-N2	85.63	66.50	79.82	73.68	86.21	64.86	98.53	86.21
	ReliefF	C	81.92	66.40	74.93	63.16	86.21	75.78	95.59	82.76
		D-F1	86.27	67.00	77.02	68.42	86.21	75.68	97.06	86.21
		D-F2	86.27	66.88	75.18	78.95	86.21	78.38	95.59	89.66
		D-N2	85.70	67.25	74.02	73.68	86.21	72.97	95.59	89.66

Table B.9: Test classification accuracy achieved by the vertical distributed approach D-F1 with overlap using C4.5 and NB algorithms, $\alpha = 0.75$.

			Isolet	Madelon	Mnist	Breast	Gli85	CLL-SUB-111	Lung cancer	11-Tumors
C4.5	CFS	5%	84.86	79.88	86.57	78.95	75.86	72.97	83.82	63.79
		10%	84.16	79.13	87.93	73.68	75.86	72.97	86.76	60.35
		15%	83.45	79.88	86.17	73.68	75.86	81.08	88.24	63.79
	INT	5%	76.52	80.25	88.70	73.68	75.86	72.97	82.35	62.07
		10%	79.28	80.25	90.71	73.68	75.86	67.57	76.47	62.07
		15%	76.72	80.25	87.77	73.68	75.86	78.38	85.29	58.62
	Cons	5%	79.03	80.25	87.06	63.16	82.76	72.97	83.82	53.45
		10%	77.42	80.25	88.72	52.63	79.31	67.57	80.88	70.69
		15%	73.83	80.25	86.22	63.16	75.86	78.38	83.82	60.34
	IG	5%	80.69	72.63	88.88	63.16	75.86	67.57	82.35	62.07
		10%	82.94	72.75	88.68	63.16	75.86	62.16	82.35	65.52
		15%	81.40	72.63	87.53	63.16	75.86	67.57	86.76	65.52
	ReliefF	5%	82.36	70.88	88.57	84.21	75.86	67.58	82.35	62.07
		10%	80.31	70.50	90.31	52.63	75.86	70.27	82.35	67.24
		15%	79.47	71.88	88.15	68.42	75.86	81.08	82.35	65.52
NB	CFS	5%	75.75	71.38	68.02	36.84	86.21	59.46	94.12	84.48
		10%	75.56	71.63	69.47	36.84	86.21	64.86	94.12	84.48
		15%	73.70	71.38	67.98	36.84	82.76	64.86	94.12	84.48
	INT	5%	66.71	70.38	73.09	36.84	86.21	72.97	94.12	84.48
		10%	72.61	69.63	72.53	36.84	79.31	70.27	94.12	86.21
		15%	65.11	70.38	75.77	36.84	82.76	78.38	94.12	87.93
	Cons	5%	65.94	70.38	71.40	36.84	72.97	72.97	89.71	77.59
		10%	72.42	69.63	74.15	36.84	75.86	70.27	88.24	81.03
		15%	51.70	70.38	69.33	36.84	75.86	75.68	89.71	82.79
	IG	5%	69.79	70.75	68.68	36.84	82.76	56.76	91.18	84.48
		10%	71.58	70.13	70.99	36.84	82.76	62.16	91.18	84.48
		15%	70.56	70.75	69.27	36.84	82.76	62.16	92.65	84.48
	ReliefF	5%	67.54	67.75	71.44	42.11	79.31	64.86	89.71	82.76
		10%	63.31	67.25	72.20	42.11	82.76	64.86	91.18	84.48
		15%	60.17	67.63	72.82	42.11	82.76	67.57	89.71	86.21

Table B.10: Test classification accuracy achieved by the vertical distributed approach D-F1 with overlap using kNN and SVM algorithms, $\alpha = 0.75$.

			Isolet	Madelon	Mnist	Breast	Gli85	CLL-SUB-111	Lung cancer	11-Tumors
kNN	CFS	5%	58.95	76.75	87.25	78.95	86.21	72.97	94.12	84.48
		10%	59.72	76.00	89.42	73.68	86.21	72.97	94.12	86.21
		15%	61.13	76.75	86.66	68.42	82.76	72.97	97.06	86.21
	INT	5%	52.98	79.00	93.56	57.89	86.21	62.16	95.59	84.48
		10%	47.92	76.63	96.39	73.68	86.21	62.16	95.59	87.93
		15%	53.11	79.00	91.38	63.16	86.42	64.86	95.59	81.03
	Cons	5%	47.59	79.00	93.41	68.42	86.21	62.16	94.12	63.79
		10%	42.14	76.63	93.61	68.42	86.21	59.46	94.12	67.24
		15%	50.35	79.00	90.65	68.42	86.21	64.86	88.24	63.79
	IG	5%	55.55	70.50	92.95	63.16	86.21	62.16	97.06	82.76
		10%	54.07	67.50	93.17	68.42	86.21	67.57	94.12	84.48
		15%	53.95	70.50	90.88	68.42	86.21	67.57	95.59	84.48
	ReliefF	5%	59.72	69.13	91.88	73.68	86.21	72.97	94.12	86.21
		10%	57.86	68.00	94.97	78.95	86.21	78.38	95.59	84.48
		15%	57.86	66.00	93.38	78.95	86.21	72.97	95.59	86.21
SVM	CFS	5%	85.63	66.63	79.16	78.95	86.21	67.57	97.06	87.93
		10%	84.67	66.75	79.68	68.42	86.21	72.97	98.53	84.48
		15%	84.54	66.63	78.28	84.21	86.21	75.68	97.06	84.48
	INT	5%	72.23	66.75	78.82	78.95	86.21	70.27	97.06	84.48
		10%	79.41	66.63	82.97	73.68	86.21	75.68	98.53	89.66
		15%	70.37	66.75	78.95	78.95	86.21	70.27	97.06	89.66
	Cons	5%	77.55	66.75	76.64	57.89	89.66	70.27	94.12	75.86
		10%	75.37	66.63	78.43	57.89	86.21	70.27	95.59	81.03
		15%	61.51	66.75	75.37	57.89	89.66	67.57	95.59	79.31
	IG	5%	85.89	66.88	79.90	68.42	82.76	70.27	97.06	86.21
		10%	85.18	66.88	79.96	63.16	86.21	67.57	97.06	84.48
		15%	83.13	66.88	79.02	68.42	82.76	67.57	98.53	84.48
	ReliefF	5%	85.44	67.88	75.80	63.16	86.21	78.38	95.59	89.66
		10%	84.67	68.38	78.33	68.42	86.21	75.68	97.06	87.93
		15%	83.98	67.75	75.60	78.95	86.21	72.97	95.59	87.93

With regard to Chapter 5, Tables B.11, B.12 and B.13 depict the classification accuracy (%) and standard deviation for the MIM, JMI and mRMR feature selection methods, respectively.

Table B.11: Classification accuracy (%) and standard deviation for MIM method.

Top features	Dataset	#Bits				
		4	8	16	32	64
5	Congress	92.64 ± 0.03	94.25 ± 0.01	94.02 ± 0.02	94.02 ± 0.02	94.02 ± 0.02
	Waveform	71.54 ± 0.01	69.46 ± 0.01	68.30 ± 0.01	68.30 ± 0.01	68.30 ± 0.01
	Connect-4	71.81 ± 0.00	69.42 ± 0.00	71.97 ± 0.00	71.97 ± 0.00	71.97 ± 0.00
	Splice	89.32 ± 0.01	88.25 ± 0.01	88.2 ± 0.01	88.25 ± 0.01	88.25 ± 0.01
	CorrAL-100	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	Led-500	79.83 ± 0.00	79.83 ± 0.00	79.83 ± 0.00	79.83 ± 0.00	79.83 ± 0.00
	GISETTE	89.65 ± 0.01	86.75 ± 0.01	84.10 ± 0.01	84.10 ± 0.01	84.10 ± 0.01
	Arcene	72.50 ± 0.04	81.00 ± 0.07	76.00 ± 0.04	76.00 ± 0.04	76.00 ± 0.04
10	Congress	91.95 ± 0.02	93.10 ± 0.02	93.10 ± 0.02	93.10 ± 0.02	93.10 ± 0.02
	Waveform	79.80 ± 0.02	78.66 ± 0.02	78.66 ± 0.02	78.66 ± 0.02	78.66 ± 0.02
	Connect-4	74.72 ± 0.01	74.32 ± 0.01	76.55 ± 0.01	76.55 ± 0.01	76.55 ± 0.01
	Splice	84.38 ± 0.01	86.30 ± 0.01	86.55 ± 0.01	86.55 ± 0.01	86.55 ± 0.01
	CorrAL-100	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	Led-500	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	GISETTE	92.17 ± 0.00	90.17 ± 0.00	90.68 ± 0.00	90.68 ± 0.00	90.68 ± 0.00
	Arcene	73.50 ± 0.05	80.00 ± 0.05	81.00 ± 0.02	81.00 ± 0.02	81.00 ± 0.02
20	Waveform	80.12 ± 0.01	79.32 ± 0.01	80.06 ± 0.01	80.06 ± 0.01	80.06 ± 0.01
	Connect-4	78.51 ± 0.01	76.56 ± 0.00	77.76 ± 0.00	77.76 ± 0.00	77.76 ± 0.00
	Splice	79.05 ± 0.02	79.09 ± 0.01	78.99 ± 0.01	78.99 ± 0.01	78.99 ± 0.01
	CorrAL-100	97.75 ± 0.00	97.81 ± 0.00	97.74 ± 0.00	97.74 ± 0.00	97.74 ± 0.00
	Led-500	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	GISETTE	93.42 ± 0.01	91.40 ± 0.01	91.47 ± 0.00	91.47 ± 0.00	91.47 ± 0.00
	Arcene	78.50 ± 0.05	81.50 ± 0.06	83.50 ± 0.04	83.50 ± 0.04	83.50 ± 0.04

Table B.12: Classification accuracy (%) and standard deviation for JMI method.

Top features	Dataset	#Bits 4	8	16	32	64
5	Congress	90.80 ± 0.00	93.79 ± 0.00	95.86 ± 0.00	95.86 ± 0.00	95.86 ± 0.00
	Waveform	60.74 ± 0.00	75.08 ± 0.00	75.26 ± 0.00	75.26 ± 0.00	75.26 ± 0.00
	Connect-4	89.10 ± 0.00	88.44 ± 0.00	88.44 ± 0.00	88.44 ± 0.00	88.44 ± 0.00
	Splice	89.10 ± 0.00	88.44 ± 0.00	88.44 ± 0.00	88.44 ± 0.00	88.44 ± 0.00
	CorrAL-100	90.62 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	Led-500	89.95 ± 0.00	89.95 ± 0.00	89.95 ± 0.00	89.95 ± 0.00	89.95 ± 0.00
	GISETTE	85.88 ± 0.00	89.23 ± 0.00	91.38 ± 0.00	91.38 ± 0.00	91.38 ± 0.00
	Arcene	78.50 ± 0.05	83.00 ± 0.05	83.00 ± 0.05	83.00 ± 0.05	83.00 ± 0.05
10	Congress	92.41 ± 0.02	95.17 ± 0.02	94.25 ± 0.02	94.25 ± 0.02	94.25 ± 0.02
	Waveform	66.04 ± 0.01	79.94 ± 0.00	79.94 ± 0.00	79.94 ± 0.00	79.94 ± 0.00
	Connect-4	71.37 ± 0.01	75.96 ± 0.01	76.49 ± 0.01	76.49 ± 0.01	76.49 ± 0.01
	Splice	85.38 ± 0.01	87.05 ± 0.01	87.02 ± 0.01	87.02 ± 0.01	87.02 ± 0.01
	CorrAL-100	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	Led-500	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	GISETTE	91.25 ± 0.00	90.28 ± 0.01	92.67 ± 0.01	92.67 ± 0.01	92.67 ± 0.01
	Arcene	79.00 ± 0.08	80.50 ± 0.11	80.50 ± 0.11	80.50 ± 0.11	80.50 ± 0.11
20	Waveform	72.84 ± 0.01	80.04 ± 0.01	79.92 ± 0.01	79.92 ± 0.01	79.92 ± 0.01
	Connect-4	77.90 ± 0.00	77.66 ± 0.00	78.41 ± 0.00	78.41 ± 0.00	78.41 ± 0.00
	Splice	80.03 ± 0.01	79.05 ± 0.01	79.34 ± 0.01	79.34 ± 0.01	79.34 ± 0.01
	CorrAL-100	97.80 ± 0.00	97.68 ± 0.00	97.76 ± 0.00	97.68 ± 0.00	97.68 ± 0.00
	Led-500	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	GISETTE	93.57 ± 0.00	93.57 ± 0.00	93.72 ± 0.00	93.72 ± 0.00	93.72 ± 0.00
	Arcene	77.50 ± 0.07	83.00 ± 0.09	83.00 ± 0.09	83.00 ± 0.09	83.00 ± 0.09

Table B.13: Classification accuracy (%) and standard deviation for mRMR method.

Top features	Dataset	#Bits 4	8	16	32	64
5	Congress	94.02 ± 0.02	94.71 ± 0.02	94.71 ± 0.02	94.71 ± 0.02	94.71 ± 0.02
	Waveform	71.24 ± 0.01	73.44 ± 0.02	76.26 ± 0.01	76.26 ± 0.01	76.26 ± 0.01
	Connect-4	69.39 ± 0.00	69.51 ± 0.00	70.74 ± 0.00	70.74 ± 0.00	70.74 ± 0.00
	Splice	87.97 ± 0.01	89.20 ± 0.01	87.99 ± 0.01	87.97 ± 0.01	87.97 ± 0.01
	CorrAL-100	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	Led-500	89.95 ± 0.00	89.95 ± 0.00	89.95 ± 0.00	89.95 ± 0.00	89.95 ± 0.00
	GISETTE	84.82 ± 1.21	87.87 ± 0.97	89.63 ± 0.72	89.63 ± 0.72	89.63 ± 0.72
	Arcene	70.50 ± 0.04	76.50 ± 0.06	76.50 ± 0.06	76.50 ± 0.06	76.50 ± 0.06
10	Congress	99.07 ± 0.03	99.05 ± 0.02	99.05 ± 0.02	99.05 ± 0.02	99.05 ± 0.02
	Waveform	77.78 ± 0.01	78.88 ± 0.016	79.74 ± 0.01	79.74 ± 0.01	79.74 ± 0.01
	Connect-4	70.84 ± 0.00	71.35 ± 0.00	72.61 ± 0.00	72.61 ± 0.00	72.61 ± 0.00
	Splice	84.50 ± 0.01	86.80 ± 0.01	86.80 ± 0.01	86.80 ± 0.015	86.80 ± 0.01
	CorrAL-100	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	Led-500	100.00 ± 0.00	100.000 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	GISETTE	89.42 ± 0.65	90.47 ± 0.52	91.65 ± 0.70	91.65 ± 0.70	91.65 ± 0.70
	Arcene	69.00 ± 0.04	75.00 ± 0.08	75.00 ± 0.08	75.00 ± 0.08	75.00 ± 0.08
20	Waveform	78.16 ± 0.01	80.08 ± 0.01	80.08 ± 0.01	80.08 ± 0.01	80.08 ± 0.01
	Connect-4	73.16 ± 0.00	71.98 ± 0.00	74.18 ± 0.00	74.18 ± 0.00	74.18 ± 0.00
	Splice	77.51 ± 0.02	79.31 ± 0.01	79.18 ± 0.01	79.18 ± 0.01	79.18 ± 0.01
	CorrAL-100	97.72 ± 0.00	97.68 ± 0.00	97.77 ± 0.00	97.77 ± 0.00	97.77 ± 0.00
	Led-500	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
	GISETTE	92.38 ± 0.64	91.50 ± 0.56	93.57 ± 0.27	93.57 ± 0.27	93.57 ± 0.27
	Arcene	73.00 ± 0.05	80.00 ± 0.06	80.00 ± 0.06	80.00 ± 0.06	80.00 ± 0.05

Appendix C

Resumen del trabajo

El *Big Data*, o datos masivos, es ya una realidad indiscutible. En los últimos años, los datos han aumentado en volumen, variedad y velocidad en varias disciplinas. Algunas estimaciones sitúan la generación de datos en 2.5 exabytes por día, creados en prácticamente cualquier campo: genómica, astronomía, experimentos en el CERN, registros de transacciones bancarias, publicaciones en redes sociales (Twitter genera 500 millones de tweets/día) o imágenes y vídeos digitales (en YouTube se suben 300 horas de vídeo cada minuto), etc. El ritmo al que en la actualidad se están almacenando datos en prácticamente todas las instituciones y empresas no tiene precedente en la historia, y ha creado la posibilidad de utilizar estos datos con el objetivo de extraer información y conocimiento, que puede ser crítico para el crecimiento o el declive de una empresa, conocimiento que podría conducir a importantes descubrimientos en ciencia, conocimiento que podría permitir identificar las causas y posibles tratamientos para enfermedades letales, etc. Por todo ello, los métodos de aprendizaje automático se han vuelto imprescindibles para extraer conocimiento útil de grandes cantidades de datos que de otra manera no tendrían sentido. En una sociedad que necesita lidiar con datos masivos, existe una necesidad urgente de desarrollar nuevas herramientas de análisis y procesamiento.

Además del aumento obvio en el tamaño de los datos, tanto en el número de muestras como en el de características, las técnicas de aprendizaje automático deben ser capaces de hacer frente también a otros desafíos dados por la gran variedad de datos, así como su naturaleza cambiante y la explosión del Internet de las cosas. En la era de los datos masivos, la complejidad de un problema de aprendizaje automático adopta múltiples formas. Algunas de esas formas son:

- Características intrínsecas de los datos. Durante las últimas décadas, se ha avanzado enormemente en el desarrollo y refinamiento de algoritmos de aprendizaje automático. Sin embargo, el aprendizaje automático todavía parece estar lejos de alcanzar ese logro en tareas diarias. Por esta razón, cuando los clasificadores no son perfectos, nos preguntamos si esto es debido a una deficiencia en el diseño de los algoritmos o a alguna dificultad intrínseca a la tarea de clasificación.

  Un requisito previo para establecer expectativas apropiadas sobre el rendimiento de la clasificación es comprender la complejidad de un problema específico que surge de una aplicación. El trabajo de Ho y Basu [60] fue fundamental para analizar la complejidad de un problema de clasificación mediante el uso de descriptores extraídos de un conjunto de datos de aprendizaje. Dado que no existe ninguna técnica de aprendizaje automático que pueda obtener el mejor rendimiento para cualquier problema de clasificación, este tipo de análisis permite determinar si existen o no patrones en los datos. También es útil obtener orientación para seleccionar técnicas de clasificación específicas. Consideramos que ésta es una de las claves para avanzar en las tareas de clasificación automática.

- Gran número de instancias. En el escenario actual regido por un aumento explosivo de los datos a nivel global, el término "Big Data" se utiliza principalmente para describir conjuntos de datos que no pueden ser percibidos, adquiridos, gestionados y procesados por las tecnologías de la información tradicionales y las herramientas de software/hardware en un tiempo razonable. Por lo tanto, este nuevo escenario ofrece oportunidades para descubrir nuevos valores, obtener una comprensión profunda de los valores ocultos y también conlleva nuevos desafíos, por ejemplo, cómo organizar y gestionar eficazmente dichos conjuntos de datos.

  La mayoría de los algoritmos de aprendizaje existentes se desarrollaron cuando los tamaños de los conjuntos de datos eran mucho más pequeños, pero hoy en día se requieren diferentes soluciones para problemas de aprendizaje a gran escala. Los problemas de aprendizaje a pequeña escala están sujetos al compromiso habitual de aproximación-estimación, pero este compromiso es más complejo en el caso de problemas de aprendizaje a gran escala, no solo debido a la precisión sino también al coste computacional del algoritmo de aprendizaje. Además, dado que la mayoría de los algoritmos se basan en estructuras de datos en memoria, estos algoritmos no son útiles cuando todo el conjunto de datos no cabe en la memoria del sistema. Esto significa que los resultados temporales tendrán que escribirse en el disco o el conjunto de datos tendrá que dividirse en particiones lo suficientemente pequeñas como para ser procesadas en memoria.

- Elevado número de características. En algunas aplicaciones, las muestras de datos están representadas por un elevado número de características. Escenarios de aprendizaje automático, como el análisis de microarrays de ADN, la clasificación de imágenes, el reconocimiento facial o la clasificación de texto, pueden tener fácilmente varios millones de características o variables de entrada, superando con creces el rango de 10-1000 considerado común hasta hace relativamente pocos años. Por ello, la llegada de datos masivos ha planteado desafíos sin precedentes a los investigadores. La alta dimensionalidad de entrada no solo implica mayores requisitos de memoria, sino también la pérdida de capacidad de generalización de los algoritmos de aprendizaje automático debido a la *maldición de la dimensionalidad*. Para evitarla, es aconsejable utilizar algoritmos de selección de características, definida ésta como el proceso de identificación de características relevantes a partir del conjunto de entrenamiento.

  Tradicionalmente, los métodos de selección de características se diseñaron para ser ejecutados en un entorno centralizado. Para hacer frente al elevado número de características fruto de la explosión del Big data, existe una fuerte demanda de métodos de selección de características escalables pero eficientes, dado que los métodos existentes no son adecuados.

- *Dataset shift*. Los sistemas basados en técnicas de aprendizaje automático a menudo se enfrentan a un importante desafío cuando estos se aplican a problemas reales. Las condiciones bajo las cuales se desarrolló el modelo pueden discernir de aquellas en las que será aplicado, debido a un sesgo intrínseco de la selección de la muestra o bien por la existencia de escenarios estacionarios. Sin embargo, la mayoría de los algoritmos de aprendizaje supervisado asumen que los datos de entrenamiento y test siguen la misma distribución de probabilidad. En problemas de clasificación, a menudo se observan cambios en el balanceo de clases. Por ejemplo, la proporción mujer-hombre es casi 50-50% en el mundo real (conjunto de test), mientras que las muestras de entrenamiento recolectadas en un laboratorio de investigación tienden a estar dominadas por datos masculinos.

Este problema complica la tarea de aprender un modelo a partir de los datos y requiere enfoques especiales, diferentes a las técnicas tradicionales, que traten las instancias que llegan con el mismo grado de importancia para el concepto final.

- Explosión del Internet de las cosas. Debido al crecimiento de la tecnología de comunicación inalámbrica y a la reducción de costes de los componentes electrónicos, el número de dispositivos relativos al Internet de las cosas se ha disparado en los últimos años. Estos dispositivos generan continuamente zettabytes de datos, que deben nutrir a un sistema de aprendizaje automático con el objetivo de analizar la información y tomar decisiones en base a ella. Sin embargo, las limitaciones en las capacidades computacionales de los sistemas embebidos portátiles—memorias pequeñas y limitado poder de cómputo—impiden la implementación de la mayoría de los algoritmos de aprendizaje automático en ellos.

  Para satisfacer esta demanda, un nuevo paradigma de computación, llamado *Edge Computing*, ha surgido. Hasta hace relativamente poco tiempo, estos millones de dispositivos que conforman el Internet de las cosas solamente registraban datos para posteriormente enviarlos a la nube, donde eran procesados para obtener información y conocimiento. El enfoque Edge Computing tiene como objetivo cambiar esta situación permitiendo que los datos sean analizados en los nodos de la red o en los propios dispositivos. De esta forma, además de evitar tráfico de red innecesario, este nuevo paradigma permite obtener conocimiento en tiempo real. También existen factores que harán que este tipo de paradigma sea menos complejo en el futuro: el coste cada vez más reducido tanto de dispositivos como de sensores se suma al aumento de su

> potencia—incluso en los dispositivos más modestos. También hay necesidades industriales que contribuyen a apostar por el enfoque Edge Computing: en ciertos entornos, la única forma de optimizar aún más los procesos es tratar de evitar la comunicación con la nube. Esto permite reducir latencias, consumir menos ancho de banda—no es necesario enviar continuamente todos los datos a la nube—y el acceso inmediato al análisis y evaluación del estado de todos esos sensores y dispositivos.
>
> Otra ventaja muy importante es la seguridad. Cuantos menos datos haya en la nube, menos vulnerable será ese entorno si se ve comprometido. Obviamente, la seguridad en estos dispositivos y sensores deberá ser atendida adecuadamente. Sin embargo, esto no significa que los entornos de computación en la nube desaparezcan: ambas tendencias deben contribuir y, por ejemplo, el enfoque Edge Computing es más apropiado cuando se necesite velocidad y baja latencia en aquellos datos que requieran una potencia de cómputo notable. Por esta razón, ambos escenarios han sido considerados en este trabajo.

En resumen, los problemas del mundo real con los que tendrán que lidiar las técnicas de aprendizaje automático tienen generalmente complejidades inherentes, como las propias características intrínsecas de los datos, su elevado volumen—tanto a nivel de muestras como características—, cambios en la distribución entre el conjunto de entrenamiento y test, etc. Todos estos aspectos son importantes y, por ello, se requieren nuevos modelos que puedan enfrentarse a estas situaciones. En esta tesis hemos abordado todos estos problemas, simplificando el proceso de aprendizaje automático, que actualmente es aún más complejo debido a la explosión del Big Data y el Internet de las cosas. En primer lugar, se realiza un análisis de complejidad para observar cómo influye ésta en la tarea de clasificación, y si es posible que la aplicación de selección de características reduzca esta complejidad. Luego, se aborda el proceso de simplificación de la fase de aprendizaje mediante la filosofía divide-y-vencerás del enfoque distribuido. A continuación, aplicamos esa misma filosofía sobre el proceso de selección de características. Finalmente, optamos por un enfoque diferente siguiendo la filosofía del Edge Computing. Esta última aproximación creemos que es pionera, pues en la literatura científica disponible en la actualidad aún no se han implementado técnicas de selección de características bajo restricciones computacionales.

Análisis de la complejidad intrínseca de los datos. Esta primera parte de la tesis está dedicada al análisis de la complejidad de los datos, la cual permite comprender si el rendimiento de un clasificador podría verse afectado, no por las limitaciones del propio algoritmo, sino por las características intrínsecas de los datos. Las características intrínsecas extraídas de los conjuntos de datos de entrenamiento en problemas de clasificación han demostrado ser predictores efectivos. Entre ellos, las medidas de complejidad de datos se pueden utilizar para identificar particularidades de los datos que implican cierta dificultad a la hora de separar las muestras del problema en sus clases esperadas, como la forma del límite de la clase, la cantidad de solapamiento entre las clases, la proximidad entre las clases o el número de muestras informativas disponibles para la etapa de entrenamiento.

Para mostrar cómo lidiar con este problema, hemos elegido como ejemplo conjuntos de datos microarray. Estos conjuntos de datos se caracterizan por su alto número de características (genes) en relación a su número de muestras, aspecto que supone un gran desafío para los investigadores dedicados al aprendizaje automático. Este tipo de datos también presenta otras particularidades que pueden afectar negativamente a la capacidad de generalización de los clasificadores, como el solapamiento entre clases y el desbalaceo de las mismas. Haciendo uso de varias medidas de complejidad de datos, se ha explorado la conexión entre la complejidad intrínseca de varios conjuntos de datos microarray (tanto binarios como con múltiples clases) y los resultados empíricos obtenidos por cuatro clasificadores ampliamente utilizados en la literatura. Además, se ha realizado un estudio para analizar si la selección de características reduce esta complejidad. Los resultados experimentales sobre 21 conjuntos de datos microarray demuestran que existe una correlación entre la complejidad intrínseca de estos conjuntos de datos y las tasas de error obtenidas por los diferentes algoritmos de clasificación.

Clasificación distribuida. En la era del Big data, los métodos de aprendizaje automático y, más específicamente, los algoritmos de minería de datos no escalan adecuadamente—requisitos de memoria y tiempos de ejecución impracticables—, dañando su rendimiento y eficiencia. Una posible estrategia consiste en distribuir la tarea de aprendizaje en varios procesadores/nodos. Así, se presenta un sistema distribuido donde las muestras de entrenamiento y test han sido extraídas de la misma distribución, con las muestras de entrenamiento distribuidas en nodos disjuntos.

Además de las tradicionales asunciones del proceso de aprendizaje, asumimos la disponibilidad de un conjunto de test lo suficientemente grande como para obtener información de su distribución. Las probabilidades de clase pueden mostrarse como un promedio ponderado de las probabilidades de clase individuales dentro de cada nodo. Estas ponderaciones dependen de las probabilidades marginales de la muestra sobre cada nodo y sobre todo el conjunto de datos. Teniendo esto en cuenta, se proponen dos enfoques diferentes para aproximar estas ponderaciones. La primera propuesta está basada en estimar la distancia entre las distribuciones de características entre cada nodo y el conjunto de test, mientras que la segunda propuesta controla la distribución de cada muestra del conjunto de test con el objetivo de minimizar estas distancias de distribución.

Por diseño, nuestro enfoque es particularmente útil en el tratamiento de nodos desbalanceados. Nuestros métodos no requieren comunicación entre los nodos, lo que permite la privacidad de los datos, la independencia del tipo de clasificador entrenado en cada nodo y la máxima velocidad en la fase de entrenamiento. De hecho, nuestros métodos no requieren reentrenamiento de los clasificadores del nodo si estos están disponibles. Resultados experimentales en varios conjuntos de datos sintéticos y reales mostraron beneficios en términos de precisión de clasificación, especialmente cuando se empleó el segundo enfoque. Además, aunque se consideraron varias reglas de combinación diferentes para agrupar los clasificadores individuales, se proporciona soporte teórico para la óptima utilización de la regla de la suma.

Selección de características distribuida. La gran explosión de datos ahora tiene el problema adicional de la gran dimensionalidad. Cuando se trata de conjuntos de datos de alta dimensión, el rendimiento de los algoritmos de aprendizaje automático puede verse degradado debido al sobreajuste, los modelos aprendidos disminuyen su interpretabilidad cuando son más complejos y, además, la velocidad y eficiencia de los algoritmos decae en relación al tamaño. El aprendizaje automático puede beneficiarse de los llamados métodos de selección de características, ya que estos son capaces de reducir la dimensión de un problema determinado. Se entiende como selección de características el proceso de detectar las características relevantes y eliminar las redundantes y/o irrelevantes, tratando de obtener un subconjunto de características lo más pequeño posible que resuelva el problema dado con una degradación mínima en el rendimiento. Sin embargo, y de manera similar a los métodos de aprendizaje, cuando se trata con grandes bases de datos, la mayoría de los algoritmos de selección de características existentes no escalan adecuadamente y su eficiencia puede deteriorarse drásticamente hasta el punto de volverse impracticable. Además, los datos a menudo se encuentran distribuidos en diferentes localizaciones, y no es ni económico ni legal reunirlos en un misma ubicación. Por estas razones, se propone una reducción de la complejidad del proceso de selección de características siguiendo el enfoque distribuido. El objetivo principal de esta estrategia es distribuir el proceso de selección de características, suponiendo que se obtendrá una reducción considerable en el tiempo de ejecución y que la precisión no se verá afectada en exceso. Así, se presentan varias propuestas para tratar tanto con una distribución horizontal como vertical de los datos. A diferencia de los procedimientos existentes para combinar las salidas parciales obtenidas de cada nodo, proponemos un proceso de fusión utilizando la complejidad teórica de estos subconjuntos de características.

Los resultados experimentales obtenidos en varios conjuntos de datos muestran que los enfoques distribuidos permiten reducir el tiempo de ejecución significativamente con respecto a la aproximación centralizada. De hecho, respecto al tiempo de ejecución, el comportamiento de las propuestas distribuidas es excelente, siendo ésta la ventaja más importante. Además, en cuanto a la precisión de clasificación, las aproximaciones distribuidas son capaces de igualar—e incluso mejorar en algunos casos—los resultados obtenidos por los algoritmos tradicionales.

Selección de características bajo restricciones computacionales. A diferencia del enfoque distribuido seguido anteriormente, aquí tratamos de reducir la complejidad de la tarea de selección de características desde la filosofía del Edge Computing. Con la llegada y estandarización de la conectividad inalámbrica y la reducción de coste de los componentes electrónicos, la cantidad y diversidad de dispositivos relativos al Internet de las cosas se ha disparado en la última década. Estos dispositivos portátiles generalmente se emplean como sistemas locales, y sus principales requisitos son trabajar con baja potencia de cómputo y pequeñas memorias. Sin embargo, estos requisitos se convierten en un gran desafío ya que estos dispositivos emergentes no son solamente sensores: deben realizar cálculos sofisticados, recopilar y agregar datos para propagarlos a la nube, y responder en tiempo real a las peticiones de los usuarios. Estos datos son la base para construir un sistema de aprendizaje automático donde se analice la información y se tomen decisiones en base a ella. Desgraciadamente, las limitaciones en la capacidades computacionales de estos dispositivos portátiles impiden la implementación de los algoritmos de aprendizaje automático tradicionales en ellos. Posteriormente, los datos deben ser enviados a una infraestructura informática remota. Sin embargo, el llamado Edge Computing ha llegado para permitirnos simplificar la tarea de aprendizaje en este nuevo escenario.

Teniendo esto en cuenta, hemos propuesto una nueva versión de la medida de información mutua—una de las medidas más comunes para el cálculo de la dependencia entre variables en los algoritmos de selección de características—utilizando en este caso parámetros de precisión reducidos. Para comprobar la efectividad del enfoque propuesto, se ha implementado esta medida en varios algoritmos de selección de características sobre un amplio paquete de conjuntos de datos sintéticos y reales. Los resultados obtenidos demuestran que representaciones bajas en el número de bits son suficientes para conseguir un rendimiento similar al obtenido mediante parámetros en doble precisión (64 bits) y, por lo tanto, abren la puerta para el uso de algoritmos de selección de características en plataformas integradas que además minimizan el consumo de energía y las emisiones de carbono, permitiendo una computación energéticamente sostenible .

Esta tesis cubre un amplio conjunto de problemas surgidos tras la llegada de los datos masivos y la explosión del Internet de las cosas. Los enfoques propuestos han demostrado su capacidad para reducir la complejidad de los métodos de aprendizaje automático tradicionales y, por lo tanto, se espera que la contribución de esta tesis abra las puertas al desarrollo de nuevos métodos de aprendizaje máquina más simples, más robustos, y más eficientes computacionalmente.