UNIVERSITY OF CASTILLA-LA MANCHA
Computing Systems Department

Preprocessing Algorithms Oriented Towards Supervised Classification of High Dimensionality DataBases: Applications to Multimedia Data Mining.

Pablo Bermejo López

UNIVERSITY OF CASTILLA-LA MANCHA
Computing Systems Department

PhD. Program:
Doctorado en Tecnologías Informáticas Avanzadas

PhD. Thesis Dissertation:

Preprocessing Algorithms Oriented Towards Supervised Classification of High Dimensionality DataBases: Applications to Multimedia Data Mining.

PhD. Student:
Pablo Bermejo López

Advisors:
José A. Gámez and José M. Puerta

A mi familia

Agradecimientos

Normalmente la sección de agradecimientos sólo interesa a los agradecidos y al que agradece. Sin embargo, tras un hito académico o personal, creo que es nuestro deber hacer un alto y hacer un leve esfuerzo por reconocer a aquellos que nos han ayudado a lo largo del camino recorrido. Por eso, me gustaría agradecer por orden cronológico a todos los que han influido positiviamente en el desarrollo de esta tesis.

Primero, a mi padre, por haberme sabido contagiar sus ganas de saber y su ilusión al verme ir creciendo académicamente hasta finalizar la carrera.

Cuando obtuve el título de Ingeniero Informático hace 5 años comencé a trabajar en la empresa privada. Durante los primeros tres meses en Madrid, me preguntaba diariamente si no me arrepentiría de no haber hecho el doctorado. Entonces, un día me llamó Luis de la Ossa y me informó sobre una convocatoria de beca pre-doctoral en el Departamento de Sistemas Inteligentes y Minería de Datos (SIMD); siempre le agradeceré el conocerme tan bien y la atención que tuvo al informarme de ello.

Después de pensármelo mucho, el factor clave para decidirme a solicitar la beca fue conocer quiénes iban a ser mis supervisores. Probablemente no la habría solicitado si no fueran José Antonio Gámez y José Miguel Puerta; así que gracias a ellos por ser quienes son y por haberme concedido la oportunidad de trabajar con ellos. Durante estos años han demostrado ser supervisores de una gran calidad humana, siempre preocupados por el presente y el futuro de los miembros del SIMD. Desde el punto de vista académico, para mí marcan dos referencias intelectuales a las que ni siquiera me atrevo a aspirar. Considero que he tenido mucha suerte de aprender de ellos y de beneficiarme de su riqueza continua en nuevas propuestas a investigar.

A lo largo de mi beca pre-doctoral (y sobretodo los 2 primeros años), Juan Luis Mateo fue, además de compañero, un tercer tutor. Probablemente estos años le haya hecho más preguntas a él, que se sentaba en mi misma sala, que a mis dos supervisores

juntos. Es imposible devolverle el favor, pero espero que sirvan mis más fervientes agradecimientos.

Al comienzo de mi tercer año realizé una estancia de 6 meses en la Universidad de Glasgow. Muchas gracias a Andrés Masegosa por hablarme del grupo MIR de dicha universidad; y también estoy enormemente agradecido a Joemon Jose por darme la oportunidad de realizar dicha estancia y esforzarse en que resultase tan fructífera como fue. Allí conocí a compañeros de gran talento y ganas de trabajar con los que he realizado una considerable cantidad de trabajos: Hideo Joho, Robert Villa, Frank Hopfgartner y Thierry Urruty.

También tengo que agradecer a Vicente López por haber ejercido su puesto en el I3A tan correctamente, siempre solventando mis problemas técnicos lo antes posible. Y a todos mis compañeros de sala por hacer el trabajo diario más ameno, y por ser cada uno especialistas en algo; investigar a su lado reduce toda complejidad a la mitad.

Fue al principio del doctorado cuando conocí a Alicia Vivo, hoy mi futura esposa. Además de haberme hecho más feliz de lo que jamás imaginé dándome a nuestro hijo Arturo, ella ha sido la única persona que ha sabido reírse de las decenas de problemas con los que me he topado a lo largo de estos 4 años. De hecho, al final me enseñó a reírme también de ellos y cuando me los encontraba incluso tenía ganas de contárselos para reirnos juntos. También, somo siempre le he dicho, nunca podré agradecerle su compañía mis primeros días en Glasgow, en los que convirtió una casa destartalda en un hogar acogedor a donde volver todas las noches después del trabajo. No sólo me das fuerzas para luchar sino también la paz que necesito.

Finalmente, de nuevo, me gustaría agradecerle a mi padre su continua preocupación por el estado de mi tesis; al igual que siempre se ha preocupado por mis estudios en el colegio, instituto y universidad. Siempre ha dicho que la mejor herencia que un padre le puede dejar a su hijo es una carrera; puede estar orgulloso, con esta tesis me ha visto dar un paso más. Sé que no hay nadie que esté más contento y orgulloso que él de que yo obtenga el grado de Doctor; y es por eso que esta tesis está principalmente dedicada a él.

Esta tesis ha sido parcialmente financiada por la Junta de Comunidades de Castilla-La Mancha (JCCM) con los proyectos PBI-05-022, PCI08-0048-8577; Ministerio de Educación y Ciencia (MEC) con los proyectos TIN1504-06204-C03-03, TIN2007-67418-C03-01; y los fondos Feder.

Acknowledgments

The acknowledgments section is usually only interesting for those who are acknowledged and for he/she who acknowledges. However, after an academic or personal milestone, I think it is our duty to make a halt and a bit of an effort in order to identify and acknowledge those people who have helped us along the way. For that reason, I would like to thank in chronological order to all those who have had an influence on the development of this thesis.

First, my father, for knowing how to transmit to me his desire for knowledge and his enthusiasm for watching how I have academically grown until finishing my degree.

When I got my degree in Computer Science Engineering 5 years ago, I started to work for a private company. During my first 3 months in Madrid, everyday I wondered if I would not ever regret not having enrolled to take a PhD. Then, one day I was called by Luis de la Ossa who informed me about an official announcement for a pre-doctoral grant in the Intelligent Systems and Data Mining Department (SIMD); I will always be grateful to him for knowing me so well and for his thoughtfulness in informing me.

After thinking about it at length, the key factor which made me decide to apply for the grant was knowing who my supervisors would be. Probably, I would not have applied for it if they were not José Antonio Gámez and José Miguel Puerta; so thanks to them for being who they are and for giving me the opportunity to work with them. During these years they have proved to be supervisors of a great human quality, always thinking of the present and future of the members from of SIMD. From the academic point of view, they represent two intellectual references to which I do not dare to aspire. I think I have been exceedingly lucky to be able to learn from them and benefit from their boundless supply of new proposals for research.

Throughout my pre-doctoral grant (and mainly during the first 2 years), Juan Luis Mateo was, besides my colleague, my third supervisor. Maybe during these years I asked this man who shared the same office as me more questions than my two supervisors together. It is impossible to ever return him the favor, so I hope my most fervent acknowledgments are enough.

In the first semester of my third year I went on a 6-month internship at Glasgow University. Many thanks to Andrés Masegosa for letting me know about the MIR group at that university; and I am also very thankful to Joemon Jose for giving me the opportunity of taking this internship and for making it so profitable. There I met colleagues of great talent and capacity for work, with who I carried out some good pieces of work: Hideo Joho, Robert Villa, Frank Hopfgartner and Thierry Urruty.

I am also grateful to Vicente López, for performing his duty at the I3A so well, always solving all my technical problems as soon as possible. Thank you also to all my office colleagues for making my daily job more pleasant; doing research next to them reduces its complexity by the half.

It was at the beginning of my grant when I met Alicia Vivo, nowadays my future wife. Besides making me happier than I had ever imagined by giving birth to our son Arturo, she is the only person who has known how to laugh at the tens of problems which I have encountered over these 4 years. In fact, she has finally taught me to laugh at them too; in fact, whenever I had a problem I could not wait for the moment to tell her and laugh together. Besides, as I have always told her, I will never be grateful enough for her company during my first days in Glasgow, during which she transformed a ramshackle house into a welcoming home to which to return at night after a whole day of work. Besides giving me strength you give me the peace I need.

Finally, and again, I would like to thank my father for his endless concern for the status of my thesis; as was the case with my studies at school, high school and college. He always said that the best inheritance a father can leave his son is a degree; he can be proud, with this thesis he has seen me take that a step further. I know there is nobody who feels happier and more proud than him for my getting the Doctoral degree; and that is why this thesis is mainly dedicated to him.

This thesis has been partially supported by Junta de Comunidades de Castilla-La

Mancha (JCCM) under projects PBI-05-022, PCI08-0048-8577; Ministerio de Educación y Ciencia (MEC) under projects TIN1504-06204-C03-03, TIN2007-67418-C03-01; and the Feder funds.

Contents

List of Figures

List of Tables

LIST OF TABLES

Part I

Introduction

Chapter 1

Introduction

1.1 Introduction

Supervised classification consists on training a classifier capable of predicting labels (from a predefined set of labels) for new instances. In general terms, the more correct labels a classifier predicts the better its performance is; thus, the training stage is crucial the construction of a good classifier. In order to improve the quality of data with which the classifier is trained, a wide range of preprocessing techniques can be found in the literature which are applied at feature or instance level: feature selection [67], feature construction [77], discretization [27], principal component analysis [55], normalization, replacing missing values [66], instance over- or under-sampling [16], clustering, . . . This thesis focuses on supervised Feature Subset Selection (FSS), Instances Selection and Instances Balancing (re-sampling).

FSS methods use a training set of instances to learn which features are the most relevant given a filter metric or some classifier. By identifying relevant features, large databases with thousands of attributes are significantly reduced, improving the performance of the classifier and helping experts to interpret the resulting model. Furthermore, unlike other reduction techniques (e.g. feature construction or principal component analysis), FSS does not alter the original representation, thus preserving the original semantics of the features.

The problem of imbalance in a dataset arises when the number of instances belonging to each of the predefined labels is very different. Thus, in the case of two possible labels {+,-} corresponding to the result of a medical test, the most common case is that only a few patients test positive, while most of them are negative. Thus, when constructing a classifier, this will be quite biased towards the negative class. So for these cases, effective over- or under-sampling techniques are mandatory in order to improve classification.

Sometimes the balance among instances of different classes is not a problem, but the presence of outliers or noisy samples is. Thus, in the same way redundant or noisy features can be discarded via feature selection, it is also possible (although it is not such a well-known approach) to perform instance selection in order to perform a more optimal learning stage for the classifier.

1.2 Structure of the Dissertation

This dissertation is structured in 4 parts.

Part I is composed of this Chapter and Chapter 2, which presents an introduction to Supervised Classification and Supervised Feature Selection. This is an important chapter since it will set out the basic principles underlying the proposals and experiments in the rest of this thesis.

Part II is composed of Chapters 3 to 6.

Chapter 3 presents several improvements to the hybrid incremental FSS algorithm IWSS: (1) a better criterion to decide the selection of a new feature in the incremental process, (2) adding the option of replacing features already selected in order to capture (in)dependences between variables; and (3) embedding the Naïve Bayes classifier in the incremental process, getting the same results but drastically reducing the complexity in time.

Chapter 4 presents a proposal to convert the deterministic IWSS algorithm into a stochastic one in order to use it in the construction stage in a GRASP search. Thus, the search space is expanded and results show that using GRASP with IWSS not only improves the performance of IWSS but it also reduces its complexity.

Chapter 5 presents a new re-ranking method applicable to incremental FSS algorithms which test the selection of features in the order indicated by a ranking based on some score of each feature with respect to the class. This proposal takes into account that some features ranked at the end of the ranking might find their score increased once some features have been selected, and so it is helpful to re-rank them to early positions and thus to stop the search before reaching the end of the ranking. Results show that this re-ranking proposal improves the performance and drastically reduces the necessary number of evaluations of several FSS algorithms over which re-ranking is applied.

Chapter 6 deals with the problem of datasets imbalance. A new family of instances re-sampling is proposed which re-samples whole new training datasets based on some distribution learned from the original training sets. Several distributions are tested and results prove that they improve the performance of Naïve Bayes Multinomial classifier applied to text categorization.

Part III is composed of two chapters dealing with multimedia-related datasets.

Chapter 7 presents an introduction to common representation features for multimedia documents, and makes comparisons between several kinds of representation to find out which perform better for the datasets used.

Chapter 8 proposes to find out if the context of users when they are performing tasks related to multimedia search affects the quality of results. Experiments suggest that instances of datasets can be selected according to the context in which they were created and thus improve the performance of classifiers.

Finally, part IV contains one chapter in which the main conclusions of this dissertation are highlighted, and possible future work is suggested. It also contains the list of publications with which this thesis has contributed to the existing literature.

1. INTRODUCTION

Chapter 2

Supervised Classification

2.1 Introduction

Classification is one of the tasks linked to Pattern Recognition, and it can be divided into Supervised Classification and Unsupervised Classification. The latter aims to discover unknown class similarities for the instances in the database, while the former is the process of predicting, as successfully as possible, the class (from a set of predefined labels) corresponding to instances which contain the same format as those instances used to learn the classifier.

In supervised classification, an instance (a.k.a. sample or object) is defined over a set of predictive variables (a.k.a. features or attributes) $X_1, X_2, \ldots, X_n$ and a class variable C which represents the class (a.k.a. label) such instance belongs to. When we store a set of instances with the same format, then we have a *database* or *corpus*, as shown in Table 2.1.

Table 2.1: Canonical Format of Databases Used in Supervised Classification.

Instance ID	X_1	X_2	$\ldots$	X_n	C
1	x_{11}	x_{12}	$\ldots$	x_{1n}	c_1
2	x_{21}	x_{22}	$\ldots$	x_{2n}	c_2
3	x_{31}	x_{32}	$\ldots$	x_{3n}	c_3
4	x_{41}	x_{42}	$\ldots$	x_{4n}	c_4
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
N	x_{N1}	x_{N2}	$\ldots$	x_{Nn}	c_N

Each predictive variable X_i can be numeric (dom(X_i)$\subseteq \mathcal{R}$) or nominal (dom($X_i = \{x_i^1, \ldots, x_i^r\}$). Each entry x_{ij} in Table 2.1 represents an attribute value pair $<X_j$, dom(X_j)$>$. Predictive attributes in the same dataset can be of different types. Some classification algorithms restrict the type of predictive attributes to use, e.g. all nominal or all numeric, or all in the same interval (a,b), so if they do not follow that format the database then needs to be preprocessed; moreover, some preprocessing algorithms need a specific format so a previous preprocessing is also needed (e.g. replacing missing values).

If C is Numeric, then the Supervised Classification task is known as Regression. In this thesis, the Class feature is assumed to be Nominal, having a predefined and finite number of possible labels.

Definition 2.1 *Given a database D following the format in Table 2.1, such that $C = \{c_1, \ldots, c_{|C|}\}$ is the set of possible labels of the class attribute, the goal of supervised classification is to build a classifier function $\mathcal{C} : x_1, x_2, \ldots, x_n \rightarrow C$ which (usually) computes a score for each label $c \in C$ and returns the label which maximizes this score.*

Definition 2.2 *The performance or goodness of a classifier $\mathcal{C} : x_1, x_2, \ldots, x_n \rightarrow C$ is usually a scalar value, where the computation of this value depends on the chosen metric and evaluation method.*

If one instance d_i can be assigned to more that one label, we are talking about *multi-label* (a.k.a. *overlapping-categories*) classification; on the contrary we refer to *single-label* (a.k.a. *non overlapping-categories*) classification. If the class can take only 2 values, this is *binomial* classification, while if it can take more than 2 values then it is a *multinomial* classification.

Moreover, Definition 2.1 refers to a *hard* classification since its returned values indicate if a given instance belongs to a class or not. On the other hand, *soft* classification returns a ranking for the class labels constructed using a computed score for which higher positions mean higher score of the instance belonging to that class label.

Furthermore, all the classification process (construction of classifier and prediction of labels for new records) may also be performed using an *automatic* process based on *machine learning* methods; *semi-automatic* if the classifier is built using feedback from the user or system; and finally, the whole process may be performed *by hand* and this requires at least one human expert who decides on a set of rules for classification.

Supervised Classification, commonly single-labeled, hard and automatic, has been applied to a wide variety of fields in the literature, for example:

- **Text Categorization (TC)** [104]. Supervised TC consists in classifying text documents into categories such as genre [29], e-mail foldering [8], news [93], . . . TC has attracted a lot of attention and it is still one of the main areas of interest in the research community for supervised classification.
- **Genomics** [76]. In the last decade, thanks to the development of automatic methods for extracting DNA samples, databases can be created containing information for DNA genes and, thus, supervised classification can be performed in order to predict an output (disease, physical property, ...) and/or to perform supervised selection of important genes to predict such a property.
- **Information Retrieval (IR)** [4; 75]. IR systems deal with text or multimedia documents (stored on a personal computer or on a network, e.g. the Internet) and their goal is to construct a ranking of documents which are relevant for the query the user has performed. IR might be performed using supervised classification methods or not; when the former is the case, then the class feature might take values “Relevant” or “Non Relevant”, and only documents labeled as “Relevant” are returned, ranked by the score metric computed.

And, in general, any application in which a previously labeled dataset is available to train the classifier and there is a need to classify new data.

2.2 Evaluation of Sup. Classification Performance

There exist a wide variety of metrics to measure the performance of the classifier and, what is more, there are also several methods to compute these metrics to avoid overfitting conclusions and to evaluate the model in a scenario as similar to its posterior real usage as possible. In this section, the most relevant validation methods and the most frequently used metrics are presented.

2.2.1 Validation Techniques

As stated in Definition 2.1, let's assume we have a classifier function $\mathcal{C} : x_1, \ldots, x_n \rightarrow C$ which returns a label $c \in C$ after evaluating an $instance$. This classifier is built from a set of instances which are previously labeled. This set of records from which the classifier is built is known as a *training set*, and it must not be used to evaluate the goodness of the learnt classifier since that would *over-fit* conclusions. For evaluation purposes, another set of labeled instances is needed, the *test set*. Then, the classifier will be run to predict, for each instance in the test set, a label and that will be compared against the real label of such instance; from the results of these comparisons, several metrics can be computed and, besides, there exist several methods to construct the *training* and *test* sets from the available dataset, that is, there exist several evaluation techniques:

- **Percentage split**. This is the simplest evaluation technique and does not have much scientific character. It consists in choosing, from the available database of labeled instances, the number of instances to be used in the building process, and the rest will be used for testing.

- **Holdout**. This method is a special case of *Percentage split*, where the training set is built using one half of the database and the other half is used for testing.

- **k-fold Cross-Validation** [79; 104]. This method is the most widely-found in the literature, and it consists of randomly splitting the dataset D into *k* disjoint splits (a.k.a. *folds*) of the same size $D = \{D_1, D_2, \ldots, D_k\}$. Then, a process is run for $i = 1, \ldots, k$: at the i-th step, fold D_i is used as test set and the union of the remaining ones as training set D^{-i}. Consequently, training sets D^{-i} share $k - 2$ folds with each other. Finally, the performance is measured as the mean of the *k* scores computed. Commonly, folds are constructed in a *stratified* manner. This means that each fold keeps the distribution of the class variable from the whole dataset D.

  The repetition of this scheme *l* times gives rise to the so-called $l \times k - fold$ *Cross-Validation*. And a common configuration for evaluation and statistical comparison purposes is 5x2cv [23].

- **Leave-one-out**. This kind of evaluation is frequently used for small datasets and it is a special case of the *k-fold CV*, in which *k* is set to the number of instances in the database... Obviously, this leads to a very computationally expensive evaluation but it provides as much training instances as possible to the classifier and, besides, it is quite realistic since, in real life, we may often need to classify new incoming instances one by one, and retrain with all the available instances again before classifying a new one.

- **Incremental time-based split validation**. As reported in [8], using training/test splits performed at random (e.g. as in standard cross validation) for classification of data with a temporal nature (e.g.: when classifying e-mails) is not appropriate because random splits may create unnatural dependencies. Because of this fact Bekkerman et al. (2005) proposed the so-called *Incremental time-based split* validation.

  This validation scheme consists of ordering instances in D based upon their $time-stamp$ field and then splitting D into subsets $\{D_1, \ldots, D_{\|D\|/z}\}$ of size $i \times z$ (except the last fold which might contain $i \times z$ plus $r < z$ instances) following the order given, where fold D_{i+1} contains instances in fold D_i besides the following z instances in time. Thus, for $i : 1, \ldots, \|D\|/z$ the classifier is trained with D_i and tested with D_{i+1}, and then we get $\|D\|/z - 1$ scores which are averaged in order to get the final performance of the classifier.

2.2.2 Scores

The scores introduced in this section are common measures used in the machine learning community to evaluate a given classifier. All these scores except "Accuracy" are computed for each possible value the class (label) may take. In addition, they all can be expressed in terms of the counts of four primary scores: *True Positives* (#TP), *False Positives* (#FP), *True Negatives* (#TN) and *False Negatives* (#FN), also computed for each possible value of the class. For example, if an instance belongs to class *c* and the classifier predicts that it belongs to that class, then this is a TP.

Commonly, predictions are summarized in a *confusion matrix*, which is a matrix of $|Class|$ rows and columns, and from which it is possible to obtain the counts of TP, FP, TN and FN.

An example of a confusion matrix is shown in Table 2.2; in this example, the class may take 4 different labels and 30 instances have been classified. The diagonal matrix shows the counts of TP for each class label, while the sum in each column except the cell belonging to the diagonal is the counts of FP, for the class value corresponding to such a column. Counts of TN for a given class label can be obtained by summing all the values in the matrix except those which are stored in a row or column representative of such a class. Finally, the counts of FN for a class label is the sum of the values in the row corresponding to such a label except the value in the column of this label. So, for class label *a*:

- #TP(a)= 6.
- #FP(a)= 0 + 1 + 2 = 3.
- #TN(a)= 7 + 2 + 1 + 0 + 3 + 0 + 0 + 0 + 5 = 18.
- #FN(a)= 2 + 0 + 1 = 3.

Table 2.2: Example of a confusion matrix with 4 possible labels for class.

		Predicted			
		a	b	c	d
	a	6	2	0	1
Real	b	0	7	2	1
	c	1	0	3	0
	d	2	0	0	5

Formally, these four primary scores can be expressed as shown in Equations 2.1 to 2.4.

$$\#TP = \#instances\ correctly\ predicted\ as\ belonging\ to\ class\ c. \tag{2.1}$$

$$\#FP = \#instances\ incorrectly\ predicted\ as\ belonging\ to\ class\ c. \tag{2.2}$$

$$\#TN = \#instances\ correctly\ predicted\ as\ not\ belonging\ to\ class\ c. \tag{2.3}$$

$$\#FN = \#instances\ incorrectly\ predicted\ as\ not\ belonging\ to\ class\ c. \qquad (2.4)$$

In terms of these primary scores, other scores can be computed: Precision, Recall, F$_\beta$-measure, E-measure, Accuracy, Precision@N, Average Precision, MAP, AUC, LIFT@N, ...

1. **Accuracy.** Accuracy (Equation 2.5) can be interpreted as the mean of precisions for all possible class labels without weighting by the number of available instances for each label.

$$Accuracy = \frac{\sum_c^{|C|} TP(c)}{\sum_c^{|C|} TP(c) + FP(c)} \qquad (2.5)$$

2. **Area Under ROC Curve (AUC).** There are also some measures which take into account the imbalance (see Chapter 6) or skewness in the dataset. This happens when a great deviation if found when comparing the number of instances for each class label. In this situation, the commonly used metric *Accuracy* is not of interest since it is quite biased toward the majority class, and the lower the cardinality of the other class the greater the problem is. For example, imagine a labeled test set with binary class and containing 100 documents, 95 of them belonging to the same class. If our classifier predicts all 100 documents belonging to the majority class, Accuracy would be 95%, while Accuracy for minority class would turn out to be 0 and, as is often the case, the class of interest is the minority class. In order to tackle this, we can use the well-known metric AUC, which is robust to this problem.

   AUC stands for “Area Under the ROC Curve”, where *ROC* stands for “Receiver operating characteristics” [26]. The ROC curve has been widely used in medical research, since it is very common in that field to use very imbalanced datasets with very few positive instances. ROC for class label *c* is obtained by plotting in the X axis the Specificity(c) or 1-Precision(c) and the Recall(c) or Sensitivity(c) in the Y axis. This curve can be interpreted as meaning that a classifier is better the nearer the hump of the curve is to coordinate (0,1); however, for comparison purposes it might be of interest to obtain a scalar value from the curve, that is, the area under the ROC curve closed by a straight line between the first point of the curve and point (1,1), as shown in Figure 2.1. Given two randomly chosen instances i1 and i2, where i1 belongs to label *c* but i2 does not, the AUC value for class label *c* can be interpreted as the probability of the evaluated classifier to tag i1 with label *c* rather than i2.

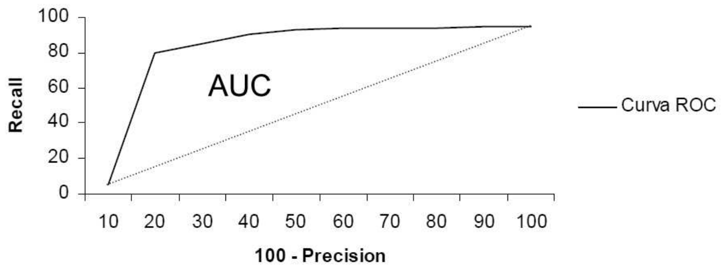

Figure 2.1: ROC curve and AUC.

3. **Precision and Recall**.

$$Precision(c) = \frac{\#TP(c)}{\#TP(c) + \#FP(c)} \tag{2.6}$$

$$Recall(c) = \frac{\#TP(c)}{\#TP(c) + \#FN(c)} \tag{2.7}$$

Precision for class label *c* can be interpreted as the probability of correctly classifying instances for class *c* without making mistakes, that is, the *exactness* of the classifier; while recall can be interpreted as the probability of correctly tagging *all* instances belonging to label *c* without giving importance to the failure in predicting the rest of the labels, that is, the *completeness* of classifier for class *c*. So, classifying all instances with class label *c* would return Recall(c)=1, but Precision would be extremely low. The natural tendency of these two measures is that one decreases as the other increases and, commonly, if they are plotted together they cross at one point called the breakeven point [54], but it is possible they never cross or that they cross more than once.

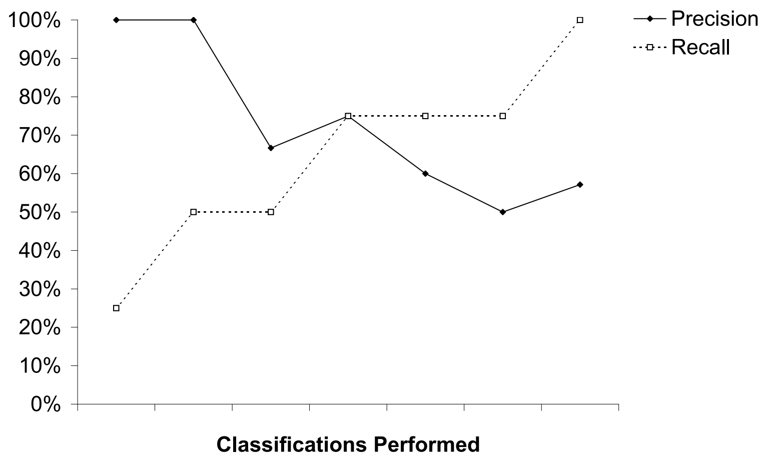

Figure 2.2: Breakeven Point example

In domains other than Machine Learning, such as medical fields, Recall is known as *sensitivity* and 1-Precision is known as *specificity*, commonly mistaken for just Precision.

4. **F_β-measure and E$_\beta$-measure.** Due to this dependency between Precision and Recall, it is often interesting to return a mean of both metrics. The most usual way to do this is with the *F_β-measure* (Equation 2.8). When $\beta = 1$ then Precision and Recall are given the same importance and this is referred as their *harmonic mean* or *F_1-measure*. If $\beta = 0.5$, precision is given twice as much importance as recall; and the contrary happens if $\beta = 2$.

$$F_\beta - measure(c) = (1+\beta^2) \times \frac{Precision(c) \times Recall(c)}{\beta^2 * Precision(c) * Recall(c)} \tag{2.8}$$

Another way to average together Precision and Recall is *E_β-measure* [92] (Equation 2.9).

$$E_\beta - measure(c) = 1 - \frac{1+\beta^2}{\frac{\beta^2}{Recall(c)} + \frac{1}{Precision(c)}} \tag{2.9}$$

5. **Lift@N** The final score to present in this Chapter is *Lift@N(c)*, which computes a ratio of the classification success in the *N* instances with highest probability (top ranked) of belonging to class *c* and the average success of the whole

database. This measure is commonly used in market contexts [9] in which the goal is to rank the most potential customers in the top of the ranking. The ranking is ordered by the probability (so it is used for soft classifiers) of belonging to class *c* (in a market context, probability of becoming a customer) and it might be split into deciles, so then *N* would refer to Lift at decile N.

An interesting comparison between AUC and Area Under Lift (AUL) is presented in [124].

$$Lift@N(c)^{1} = \frac{Precision(c)@N}{Precision(c)@|test\ set|} \tag{2.10}$$

6. **AveragePrecision and MAP** Precision is a most important metric for Information Retrieval systems. If the returned ranking is too long, it is more probable that some documents classified as “Relevant” are indeed “Non Relevant” and thus Precision decreases[2]. Thus, it is important to compute how good (in precision terms) the ranking algorithm is at a certain length N of classified documents, where N stands for “top N documents in ranking”. The goal is to make sure that the top ranked documents give a high precision.
The main problem of Precision@N is that it does not provide any information about Recall. Maybe for enormous databases such as the Internet, recall is not very important but that is not the case for personal or enterprise databases. In order to tackle this problem, *AveragePrecision* can be used.

$$AvgP = \frac{\sum_{n=1}^{N} Precision@n \times isRelevant(nth\ document)}{Relevant\ documents\ in\ whole\ database} \tag{2.11}$$

The denominator in Equation 2.11 adds recall influence to the formula, but it does not add any punishment when the ranking contains “Non Relevant” documents; and the same happens when computing *MAP* (Mean Average Precision) which is the mean of computing *AvgP* for several queries.

[1]As it happens in IR, class in market business is usually binomial (“Customer buys”, “Customer does not buy”), so the usual notation for Lift@N(c) is just Lift@N, referring to the positive class label.

[2]When the class is binary, it is not usual to denote metrics as, for example, Precision(c), but just Precision, which refers to Precision of the positive class. In the IR case, it refers to Precision of “Relevant” documents.

The comparison of classifiers using metrics can sometimes be biased by just the nature of the classifier; thus, [11], performed a study to find out what metrics suit best depending on the classifier used, using the same databases and evaluation method. Some of those metrics have been presented here and others have not, but the conclusion is that one classifier might be better than another due to the metric and not just to the classifier itself. Besides, it is well known that the nature of the database might also bias the comparison; for example, an imbalanced dataset classified with a classifier which is non-robust against skewness will perform worse, in terms of Accuracy, than when the same classifier is run over the same dataset but filtered through a balancing process.

So, as a consequence, a correct methodology is important at the time of comparison or inferring conclusions. Thus, it is necessary to explain why a given metric has been chosen and why the applied evaluation method is appropriate. Moreover, comparisons should not be made based only on one database but several in order to make correct statistical comparisons.

2.3 Algorithms for Supervised Classification

In the literature we can find a vast number of classifiers of different nature which are used for supervised classification; that is, to predict the Class label given an instance **X** composed of predictive features **X** $= \{X_1, \ldots, X_n\}$.

This section introduces some of the most important supervised classifiers from different families:

- **Naive Bayes**, **Semi-Naive Bayes** and **TAN**: these are Bayesian classifiers based on the Bayes Theorem.
- **Neighbourhood-based classifiers**: a vicinity-based classifier.
- **c.45** and **ID3**: decision tree classifiers.
- **Neural Network**: based on a set of neurons connected to each other and working as black boxes.
- **Support Vector Machines**: one of the most important kernel-based methods.

2.3.1 Bayesian Classifiers

Bayesian classifiers [24] are methods that are used to perform supervised classification. They are a Bayesian network in which each node except the Class C node represents a predictive feature X_i, and the goal is to compute the probability for each possible c value (label) of node C using Bayes's Theorem as shown in Equation 2.12. Then, the label c which maximizes this probability *a posteriori* (Equation 2.13) is returned as output.

$$p(c|x_i, ..., x_n) = \frac{p(c)p(x_1, ..., p_n|c)}{p(x_1, ..., x_n)} \propto p(c)p(x_1, ..., p_n|c) \tag{2.12}$$

$$\begin{aligned} c_{MAP} &= \underset{c\in\Omega_C}{\operatorname{argmax}}\, p(c|x_1, ..., x_n) && (2.13)\\ &= \underset{c\in\Omega_C}{\operatorname{argmax}} \frac{p(x_1, ..., x_n|c)p(c)}{p(x_1, ..., x_n)} && (2.14)\\ &= \underset{c\in\Omega_C}{\operatorname{argmax}}\, p(x_1, ..., x_n|c)p(c) && (2.15)\end{aligned}$$

- **Naïve Bayes (NB)** Computations in Equation 2.12 involve managing distribution tables with many features dependent upon each other. In order to avoid this complexity, a series of alleviations have appeared in the literature.

  The strongest alleviation is that which makes the assumption of independence among all predictive features given the class, this model is known as Naïve Bayes (NB) and its structure is shown in Figure 2.3.

  As a consequence of this independence assumption, computations required by Equation 2.13 can be simplified to Equation 2.16.

$$c_{MAP} = \underset{c\in\Omega_C}{\operatorname{argmax}}\, p(x_1, ..., x_n|c)p(c) = \underset{c\in\Omega_C}{\operatorname{argmax}}\, p(c)\prod_{i=1}^{n} p(x_i|c) \tag{2.16}$$

  Although it may seem that the independence assumption among predictive features is too strong, it is possible to find in the literature many cases in which the performance of NB classifiers has been as good as (or even better than) more

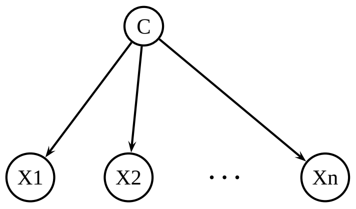

Figure 2.3: Structure of Naïve Bayes Classifier.

complex methods [82]. But, on the other hand, there are other cases in which NB performs worse. Thus, some methods exist to add complexity to the NB scheme [38], resulting in new classifiers: *Augmented Networks*, *k-dependence Bayesian and Semi-NB* classifiers.

- **Augmented Networks** Augmented Networks are similar to the NB structure but with the addition of the possibility for predictive features X_i to have as father another predictive feature X_j besides the class C; this link between X_i and X_j represents dependency between them.

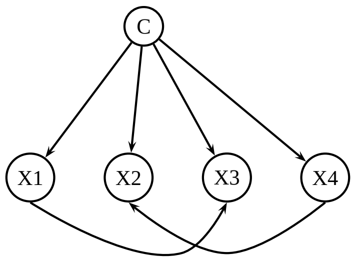

Figure 2.4: Example of an Augmented Classifier.

Figure 2.4 shows the structure of an Augmented Network; for this example, the c_{MAP} would be computed as shown in Equation 2.17.

$$c_{MAP} = \underset{c \in \Omega_C}{\operatorname{argmax}} \, p(c|x_1, ..., x_n)p(c) \tag{2.17}$$
$$= p(c)p(x_1|c)p(x_2|x_4, c)p(x3|x_1, c)p(x_4|c) \tag{2.18}$$

- **Tree Augmented Network (TAN).** Finding the best Augmented Network is an NP-hard problem; thus, [38] proposes to add the restriction to this scheme of having a tree structure (TAN). Another proposal can be found in [82], which focuses on finding the augmented structure with highest classification performance. Finally, [38] and [85] compute the mutual information between two predictive features given the class in order to increase the likelihood.
- **kDB.** kDB structures [100] are generalizations of TAN, where predictive features are limited to *k* predictive features as parents, besides the class. So TAN is a kDB structure with k=1.

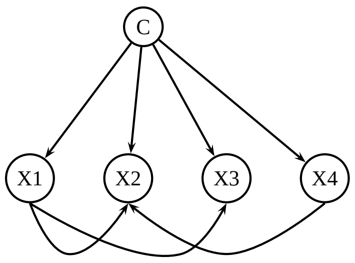

Figure 2.5: Example of kDB structure with k=2

Figure 2.5 shows an example on a kDB structure with k=2. In this case, the c_{MAP} value is computed as shown in Equation 2.19

$$c_{MAP} = \underset{c \in \Omega_C}{\operatorname{argmax}} \, p(c|x_1, ..., x_n) \tag{2.19}$$
$$= p(c)p(x_1|c)p(x_2|x_1, x_4, c)p(x3|x_1, c)p(x_4|c) \tag{2.20}$$

- **Semi-Naïve Bayes (SNB).** SNB structures are identical to that of NB, but the difference is found at the nodes, which can be *atomic* or *joint*. *Atomic* nodes are ordinary nodes representing one predictive feature X_i, while *joint* nodes represent two or more predictive features and, thus, a statistical relation between them.

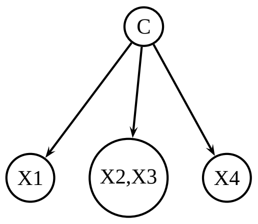

Figure 2.6: Example of SNB structure.

Since the structure is the same as in NB, the c_{MAP} is also computed in the same way (Equation 2.16). Then, the most important part in SNB structures is how to construct nodes.

When working with nominal predictive features, states of a joint node are the result of the Cartesian product of the states of the predictive features which compose it [82]. Pazzani presents two algorithms to construct an SNB structure: *forward sequential selection and joining* and *backward sequential elimination and joining*. The former starts with just the class feature C added to the classifier, and the latter starts with an SNB structure with all the features. In both cases, at each step the selected operation over the network is that which maximizes the performance of the classifier (wrapper approach). In the *forward* method, possible operations are:

1. Add a predictive feature not yet added, making it dependent on the class and non-dependent (given the class) on the rest of added predictive features.
2. Join a predictive feature not yet added with a predictive feature previously added.

While in the *backward* method, the possible operations are:

1. Remove a predictive feature from the structure (atomic or joint).
2. Join a predictive feature in the structure with another predictive feature also in the structure.

2.3.2 Classification Trees

Classification Trees (Figure 2.7) are models based on a recursive partition method which aims to divide the dataset using only one predictive feature at each level of the tree structure. A decision tree can be used to visually represent a set of rules for decision making. At each level of a tree, one predictive feature is represented by a node, and this node has as many branches as the number of possible values this feature may take. Then, leaf nodes represent labels of the class feature; these labels may be repeated across the same or different levels, since different rules may lead to the same classification.

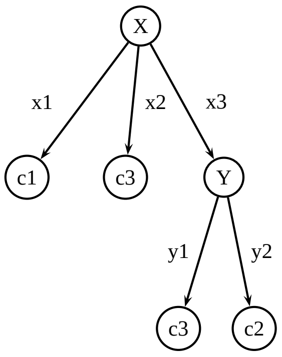

Figure 2.7: Example of a Classification Tree structure.

The predictive feature at each level is selected based on a given criterion or algorithm; then, depending on the algorithm used we have different classification tree models, the most widely known being ID3, and its improved version C4.5 [90]. The former uses as criterion for selecting a feature at each level the Info Gain metric (Section 2.4.1) and the latter uses Info Gain Ratio (Section 2.4.1). Besides node selection, tree construction algorithms need to specify the stop criterion and the post-pruning process to simplify the tree structure.

2.3.3 Classification algorithms based on proximity.

Classification algorithms based on proximity [22] are theoretically simple and have a long background in supervised classification. Instances whose class is known are represented in a vectorial space, where each coordinate *i* corresponds to the *i-th* attribute used in the representation of the instances.

These algorithms are called lazy because there does not exist a training step to build a model which will be used to classify the instances. So each time a new instance is to be assigned a class it will be necessary, in the most general and basic case, to compute the distance of this instance to all the other instances in the database. Thus, the NN (*Nearest Neighbor*) algorithm would assign to this instance the label of the nearest one to it in the vectorial space.

From this NN algorithm, several algorithms based on neighborhood have been developed. The straight evolution from NN is the K-NN algorithm, in which the assigned class is the one which gets more votes among the K nearest neighbors. In this way erroneous assignments are avoided in those cases where the closest instance does not belong to the same class the new instance belongs to (spatial intrusion reduction or noise). Among the many variants of K-NN we can summarize:

- *Different weights for attributes* [19]. When computing the distance between two instances the value of each dimension of the instances is used. Since dimensions correspond to attributes used to represent the instance, it seems a good idea to give more weight to those attributes which are more relevant than others. A way to decide this importance might be a prior selection or ranking of the set of attributes.

- *Use of a voting threshold*. A minimum number of votes can be established that a class should receive before assigning it to the instance being classified. If no class receives votes over this threshold the instance remains unclassified.

- *Average distance*. K nearest neighbors are not used to vote, but average distance is computed for the instance being classified to each of the K neighbors belonging to the same class. Thus the assigned class is the one which minimizes the average.

- *Use of centroids*. The K-NCN [102] algorithm selects, among all instances in the training set, the instance or centroid representative of each class. Then, distances are not computed to each training instance but to each centroid. This approach drastically reduces the computational load, but it makes the centroid selection a very important phase for a successful classification.

There exist two key decisions to make when designing a classification algorithm based on proximity: the representation of dimensions (attributes) in samples and the distance metric used to compute distances between two samples. With regard to the distance metric, the most common are those belonging to the *Minkowski* distances family [15] (such as the Euclidean Distance).

2.3.4 Other classifiers: Support Vector Machines (SVM) and Artificial Neural Networks (ANN).

SVMs [10] are originally linear classifiers which aim to find the optimal (set of) hyperplane(s) to separate instances in the training set such that each separated group of instances is tagged with the same class label. There may be several hyperplanes separating classes, but the best hyperplane (*maximum-margin hyperplane*) is that which maximizes the distance between the nearest instances of such groups.

In [10] several new kernels are presented so that SVMs can be used as non-linear classifiers.

ANNs simulate properties of observed biological neural systems using statistical methods. Figure 2.8 shows a general structure of ANN, which consists of simple elements called *neurons* connected to each other, as well as being connected to the input and output of the structure. These connections are called *synapses* and there is a weight associated to each of them. Each neuron can be seen as a black box, which receives an input through one or more weighted connections and produces a non-linear output. Commonly, the most important parameters to tune in an ANN are: number and layout of neurons, non-linear activation functions and synapse weighting

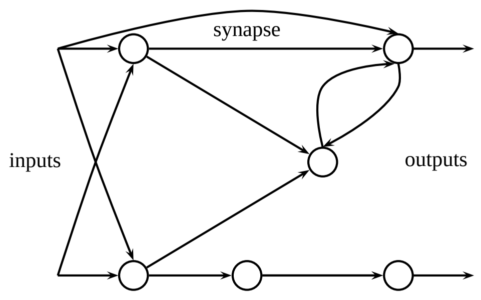

Figure 2.8: General structure of a Neural Network.

2.4 Supervised Feature Selection

Feature (or variable, or attribute) Subset Selection (FSS) is the process of identifying the input variables which are relevant to a particular learning (or data mining) problem [41; 67]. Though FSS is of interest in both supervised and unsupervised data mining, this Chapter deals with supervised learning, and in particular with the classification task. Classification-oriented FSS carries out the task of removing most irrelevant and redundant features from the data with respect to the class. This process helps to improve the performance of the learned models because it:

1. Alleviates the effect of the curse of dimensionality.
2. Increases the generalization power.
3. Speeds up the learning and inference process.
4. Improves model understandability

Furthermore, unlike other reduction techniques (e.g. feature construction [67; 77] or principal component analysis [55]), FSS does not alter the original representation, so it preserves the original semantics of the variables, helping domain experts to acquire better understanding about their data by telling them which are the important features and how they are related with the class.

The general process of FSS can be described by: (1) a Search Method and (2) an Evaluation of Feature subset Method.

There are several kinds of search methods ([70]), the most important being: *Ranker*, *Exhaustive*, *Sequential*, and *Metahueristics*. On the other hand, Evaluation may be: *filter, wrapper* or *hybrid*.

Besides this classification, there also exist *Embedded FSS methods*, which consist of using classifiers which select by themselves the subset of variables they need, as happens with the C4.5 construction algorithm.

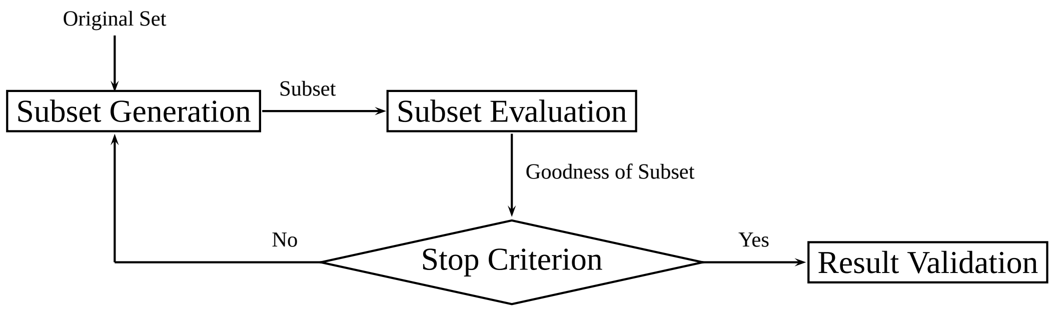

Figure 2.9: General FSS process.

Evaluation methods:

1. *Filter*. Filter techniques are those that evaluate the goodness of an attribute or set of attributes by using only intrinsic properties of the data (e.g. statistical or information-based measures). Filter techniques have the advantage of being fast and general, in the sense that the resultant subset is not biased in favor of a specific classifier. Examples of frequently-used filter algorithms are Relief [59], FCBF [128] and different approaches based on the use of mutual information [6; 31; 69].

2. *Wrapper*. On the other hand, wrapper algorithms are those that use a classifier (usually the one to be used later for test or validation) as a surrogate in order to assess the quality of the attribute subset proposed as candidate by the search

algorithm. Wrapper algorithms have the advantage of achieving a greater accuracy than filters but with the disadvantage of being (far) more time consuming and obtaining an attribute subset that is biased toward the classifier used.

3. *Hybrid*. Recently a new family of sequential search methods have arisen which combine filter and wrapper evaluations [32; 98] in order to take advantage of both techniques.

2.4.1 Filter Subset Evaluation

There exist a large number of filter metrics which compute the predictive power of one or more features with respect to the class (filter metrics for supervised FSS). The main advantage of these metrics is the low computational complexity compared to wrapper approaches.

The choice of the filter metric to use depends, among other things, on the performance metric to be used for the classifier. For example, Information Gain (IG) is reported in [34] to be the best choice if the goal is Precision.

Evaluation may be *univariate* for single feature evaluation or *multivariate* if the selection of a new candidate feature is evaluated using the currently selected features subset. The main problem of univariate evaluations is that they cannot detect conditional dependences or independences. Thus, single features marginally relevant to the class are directly selected by univariate methods, and there is no check to see if they are not relevant given that another features are currently selected. In this way, it is possible that the addition of a marginally relevant feature may decrease the goodness of the current subset selected instead of increasing it. Some of the best-known filter metrics are based on Shannon Entropy (Equation 2.21), as shown here.

$$Entropy(X) = H(X) = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i) \tag{2.21}$$

- **Info Gain (IG)**. This metric measures the change in entropy for class feature C once a predictive feature X is known.

$$IG(C, X) = H(C) - H(C|X) \tag{2.22}$$

- **Info-Gain Ratio (IGR)**. It penalizes IG when the predictive feature has many states.

$$IGR(C,X) = \frac{IG(C,X)}{H(X)} \tag{2.23}$$

- **Symmetrical Uncertainty (SU)**. SU is a nonlinear information theory-based measure that can be interpreted as a sort of Mutual Information normalized to interval [0,1].

$$SU(C,X) = 2 \times \frac{IG(C,X)}{H(C)+H(X)} \tag{2.24}$$

- **Chi-Squared**. This is not just a metric but a statistical test, which in this case can be used to evaluate the value of the chi-squared statistic with respect to the class, using as null hypothesis that feature *X* is independent of the class.

- **Correlation-based Feature Selection (CFS)**. This metric evaluates sets of predictive features in terms of SU between pairs of them and SU between each of them and the class feature.

$$CFS(X_1,\ldots,X_n) = \frac{\sum_{i=1}^{n} SU(C,X_i)}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} SU(X_i,X_j)}} \tag{2.25}$$

- **Conditional IG-Battiti**. Battiti [6] approximates the computation of IG for a candidate feature X_i given the already selected features subset S' as shown in Equation 2.26, where β is usually set to 0.5.

$$IG(X_i,C|S') = IG(X_i,C) - \beta \sum_{s\in S'} IG(X_i,s) \tag{2.26}$$

- **Conditional IG-Peng**. Peng et al. [84] uses another approximation for the computation of IG for a candidate feature X_i given the already selected features subset S'.

$$IG(X_i,C|S') = IG(X_i,C) - \frac{1}{|S'|} \sum_{s\in S'} IG(X_i,s) \tag{2.27}$$

2.4.2 Search Methods

Following the taxonomies proposed in [70] and [47] we can distinguish between complete, deterministic (sequential) heuristic and non-deterministic (stochastic) heuristic algorithms. In terms of complete algorithms we can consider algorithms which evaluate all possible subsets by following depth-first and breadth-first search strategies [68], and the branch-and-bound algorithm proposed in [109]. The *deterministic heuristic* approach refers to the family of methods that include forward and backward greedy sequential algorithms [60], floating selection algorithms [80; 89] and best-first search or hill climbing [62]. On the other hand, *non-deterministic heuristic* methods use randomness in order to avoid getting stuck in local optima. Examples of this approach are estimation of distribution algorithms (EDAs) [47] and genetic algorithms (GAs) [126].

Since complete searches are usually too expensive, stochastic or sequential searches are preferred. Usually in these searches, the more the solutions space is explored, the better the performance achieved is.

This thesis contains methods for deterministic sequential (Chapters 3 and 5) and random sequential (Chapter 4) FSS.

Some of the most important filter and wrapper search methods are:

- **Fast Correlation-Based Filter (FCBF) search.** FCBF [127] is a ranker search method with two steps: (1) based on a given threshold, it selects features from a filter ranking constructed with a univariate evaluation by Symmetrical Uncertainty of features with respect to the class; and (2) based on three heuristics, it removes features highly correlated between pairs from the set of selected features from the previous ranking. These heuristics avoid the evaluation of SU between each pair of features, and thus the FCBF search proves to be a fast filter method at the time that it avoids the selection of redundant features.

- **Sequential Forward Selection (SFS)**. SFS [41] starts with a subset of selected features $S' = \emptyset$, and performs a forward greedy selection with (usually) wrapper evaluation at each step. When no addition of any feature to S' improves the performance, then the stop criterion is triggered and the search is finished.

- **Mutual Information-based Feature Selection (MIFS)**. MIFS [6] is a forward greedy search which uses the filter evaluation shown in Equation 2.26. Starting with subset S containing the feature X_i which maximizes $IG(X;C)$, the following features added are those which maximize Equation 2.26. The stop criterion is fulfilled when the cardinality of the final subset of selected features reaches a pre-defined value *k*.

- **Max-Relevance and Min-Redundancy (mRMR)**. mRMR [84] is also a forward greedy search and also starts subset S' containing the feature X_i which maximizes $IG(X;C)$, but the filter evaluation is the one shown in Equation 2.27. The stop criterion is the same as in MIFS.

- **Conditional Mutual Information Maximization (CMIM).** CMIM [31] is a forward greedy search with the same start for subset S' as MIFS and mRMR. The incremental addition of the following feature is decided as the feature X_i not yet selected which maximizes the information $I(X_i;C|s_j)$, where s_j is the feature $s_j \in S'$ which minimizes that information (Equation 2.28).

$$\forall X_i \in S, \forall s_j \in S', add\ X_i\ such\ that\ \underset{Xi}{\operatorname{argmax}}\{\min_{sj} I(X_i;C|s_j)\} \tag{2.28}$$

The idea behind MIFS, mRMR and CMIM is to take into account features already selected when regarding a new feature X_i to be added to the final subset S' of selected features; this would mean computing conditional information $I(X_i \wedge C \wedge S')$. As cardinality in S' grows throughout the incremental process, computation of this probability becomes unfeasible, and thus MIFS, mRMR and CMIM provide heuristic approximations in order to simplify the computations as long as that conditional relevance is still detected. Chapter 5 introduces a new FSS method based on this idea.

As mentioned above, there exists a new family of hybrid search methods which combine filter and wrapper evaluations to guide the search. Some well-known sequential searches with hybrid evaluation are:

- **Linear Forward Selection (LFS).** LFS [40] is a simple complexity optimization of SFS. It consists of first creating a filter ranking based on SU and selection of the *k* first features; then, the SFS algorithm is run over the selected features.

- **Best Incremental Ranked Subset for Feature Selection (BIRS).** The BIRS [98] algorithm first produces a filter ranking and then it performs an incremental best-first selection throughout the ranking. The inclusion of each feature in ranking is tested by adding it to S' and performing a wrapper evaluation using all features in S'. Moreover, BIRS provides a heuristic rule in order to decide if the new feature must be kept in S' when it provides better performance than before adding it.

- **Best Agglomerative Ranked Subset for Feature Selection (BARS).** BARS [97] is based on a heuristic rule that obtains good subsets by iterating between two phases: (a) ranking of subsets and (b) generation of new candidate subsets by combining (based on wrapper evaluation) those previously ranked. The heuristic nature of the algorithm lets it evaluate a reduced number of candidate subsets and its outstanding characteristic is that it obtains very compact subsets.

2.5 Summary

The task of supervised classification has been presented. It provides several evaluation methods and metrics in order to measure the goodness of a classifier constructed with the available tagged data. This goodness gives us an idea of how well the classifier would perform when predicting the tag of new unlabeled data, that is, in a future scenario where the classifier would be trained with available data and then applied on new instances whose tag is unknown and this is what we want to automatically predict.

Besides this, several methods for evaluating the standard classifiers and a taxonomy of feature subset selection techniques have been presented. As explained, feature selection reduces the dimensionality of the database and it may help the classifier to perform better.

Part II

Preprocessing Algorithms in Supervised Classification

Chapter 3

Incremental Wrapper Selection

3.1 Summary

As mentioned in Chapter 2, wrapper techniques used to perform better than filter methods. However, wrapper searches are much slower and, with the venue of this decade, new databases have appeared which contain thousands or tens of thousands attributes, making traditional wrapper algorithms unfeasible. Recently, a new family of hybrid search methods have arisen which combine filter and wrapper evaluations in order to guide the search, try to benefit from the advantages of both approaches.

This chapter describes the Incremental Wrapper Subset Selection (IWSS) algorithm, which is a hybrid search with two phases: (1) creation of a filter ranking and (2) performing a sequential forward search with wrapper evaluations. This chapter presents four contributions to the IWSS algorithm:

- (*Relevance Criterion*). First, several Heuristic criteria are studied in order to decide when to accept or reject a new feature along the sequential process.

- (*Replacement*) Second, it is proposed a method to improve the subset selection process in order to obtain even more compact subsets. The idea is that when a new attribute is being analyzed, we test not only the possibility of adding it to the selected subset, but also swapping it with any of the already included attributes.

- (*Early stopping*) Third, it is studied the impact of reducing the number of wrapper evaluations by stopping the process before analyzing all the variables in the ranking.

- (*Embedding*) And last, it is shown how for some classifiers (that can be incrementally updated by progressively adding attributes) the model can be embedded in the selection process, obtaining in this way a (really) significant reduction in the CPU time needed.

3.2 Introduction to IWSS

The idea behind IWSS is to use a filter measure in order to obtain a ranking of the attributes' relevance with respect to the class. Then, a sequential algorithm is used to run over the ranking by incrementally adding those variables that are relevant to the classification process, where the relevance of including a new variable is measured in a wrapper way. The main advantage of this approach is that it retains a great part of wrapper advantages, while reducing the computational cost to $\mathcal{O}(n)$ wrapper evaluations instead of $\mathcal{O}(n^2)$ as happens with pure wrapper approaches. When dealing with thousands of variables this point makes the difference between considering the task computationally feasible or not.

An interesting contribution to *Incremental Wrapper-based FSS* (IWSS) is introduced in [98]. In that paper the authors propose the use of a relevance criterion in order to decide when a new attribute must be included in the selected subset. The relevance criterion is based on a t-test instead of just comparing the mean accuracy, and the results reported show that the use of this relevance criterion frees the algorithm from noise and so more compact subsets can be obtained with similar (statistically non-different) accuracy. In Section 3.3, a deeper study of the relevance criterion is carried out by using different significance levels in the t-test, and it is also considered another statistical test (Wilcoxon signed rank test) and a simple heuristic criterion.

Let's describe as the canonical IWSS algorithm the one shown in Figure 3.1.

As it can be seen, steps 1 to 4 are devoted to computing the ranking. This stage requires $\mathcal{O}(n)$ filter evaluations. As in other works [32; 98], Symmetrical Uncertainty (SU) (see Section 2.4.1) is used to evaluate the predictive attributes. Attributes are ranked in increasing SU order; that is, more informative attributes are placed first.

```
In    $\mathbf{T}$ training, $M$ measure, $\mathcal{C}$ classifier
Out   $\mathcal{S}$    // The selected subset
1     list $R$ = {}    // The ranking
2     for each attribute $A_i \in \mathbf{T}$
3       *Score*=$M_{\mathbf{T}}(A_i$, class)
4       insert $A_i$ in $R$ according to *Score*
5     $\mathcal{S}$ = $\{R[1]\}$
6     BestData = evaluate$(\mathcal{C}, \mathcal{S}_{aux}, \mathbf{T})$
7     for i = 2 to $n$    // $n = R.size()$
8       $\mathcal{S}_{aux}$ = $\mathcal{S} \cup \{R[i]\}$
9       AuxData = evaluate$(\mathcal{C}, \mathcal{S}_{aux}, \mathbf{T})$
10      if (AuxData $\triangleright$ BestData)
11        $\mathcal{S}$ = $\mathcal{S}_{aux}$
12        BestData = AuxData
```

Figure 3.1: IWSS canonical algorithm.

Steps 5 and 6 carry out the initialization of $\mathcal{S}$ by using the first variable in the ranking. Also the data resulting from evaluating that subset are stored in BestData. In particular, it is assumed that function $evaluate(\mathcal{C}, \mathcal{S}_{aux}, \mathbf{T})$ learns and validates the classifier $\mathcal{C}$ by using a 5-fold cross validation over the training set $\mathbf{T}$ projected over subset $\mathcal{S} \cup \{C\}$. Thus, BestData will contain an array BestData.f[1..5] with the accuracy obtained for each fold and a real value BestData.av with the averaged accuracy over the 5 folds.

Steps 7 to 12 carry out the main cycle of IWSS. Depending on the way step 10, or to be more precise operation $\triangleright$, is implemented we get different IWSS algorithms. Thus, if (AuxData $\triangleright$ BestData) is implemented as (AuxData.av $>$ BestData.av) we get IWSS with the simplest relevance criterion; if (AuxData $\triangleright$ BestData) is implemented as follows:

> if (AuxData.av $>$ BestData.av) and
> (test(t-test,$\alpha = 0.1$,BestData.f[],AuxData.f[]))

then we get the BIRS algorithm [98].

In Section 3.3, a deep study is performed in order to compare different relevance criterion for operation $\triangleright$. Best relevance criterion found in those experiments (*MinFoldersBetter* criterion) will be the default instantiation of operation $\triangleright$ in the following executions of IWSS in this chapter.

3.3 Relevance Criteria in IWSS

As reported in [98], using the relevance criterion based on statistical testing prevents the algorithm from including new variables due to noise or outliers (e.g. only in one of k folds in cross-validation is the accuracy significantly higher). On the other hand, even based on statistical hypothesis testing, these relevance criteria are of a heuristic nature because of the small number of samples (5), which forces the selection of non standard values for α.

3.3.1 Three criteria for relevance decision of new features.

Experiments in [98] only use the t-test with $\alpha = 0.1$ in order to instance operation $\triangleright$ from Figure 3.1. Since this proves to perform better than just accepting new attributes which improve performance with any value, it is interesting to make a deeper and breadther research at this respect. Thus, in the following three criteria are proposed to be tested with different restrictive levels:

Criterion 1. Student t-test.- This considers the relevance criterion proposed in [98]; that is, a parametric statistical test: *paired two-tailed Student's test.* This is likely one of the most used test in machine learning statistical analysis, however it assumes a Gaussian distribution over the paired differences between the two datasets, which is not always satisfied. This is the case here because of the small sample size (5 folds), but as Ruiz et al. [98] point out the goal is to have an objective criterion about the relevance of including a new feature, not to make a statistical analysis of the populations. Another known problem of this test is that it is affected by outliers. In [98] the authors set $\alpha = 0.1$, so here experiments will be performed to study the effect of using less restrictive α values (0.1, 0.15, 0.2 and 0.25).

Criterion 2. Wilcoxon signed-ranks test [121]. With the same idea of using an objective criterion as the described in the previous paragraph but avoiding the Gaussianity assumption, this criterion is the use of a non-parametric test. In this case Wilcoxon signed-ranks test is selected because it is one of the most frequently used in the machine learning literature. In [23] a expression to compute the *z* statistic value is provided for large sample size cases (e.g. > 25) but, since this is not the case, exact *z* statistic values are used (which can be found in many statistics books) for given alpha values and samples size in these experiments. This test is not so affected by outliers (as the t-test) since is checks values of paired differences instead of values of each sample. This is a rather important advantage specially in the case of having really few samples. As in the previous case experiments are are run with α=0.1, 0.15, 0.2 and 0.25.

Criterion 3. Minimum better folds heuristic. In this case the goal is to test a pure heuristic criterion that tries to reject the same null hypothesis than in the previous cases in favor of the same alternative hypothesis, e.g., the mean of the values in set AuxData.f[] is significantly different to mean of values BestData.f[]. Thus, with the idea of avoiding to include a new feature because a noisy result, it is imposed that the inclusion of a new variable in $\mathcal{S}$ is allowed only when the following comparison holds:

if (AuxData.av $>$ BestData.av) and #(BestData.f[] $>$ AuxData.f[]) $\geq$ mf

where mf is the minimum number of folders in which it is required $\mathcal{S}_{aux}$ to be better than $\mathcal{S}$. Experiments are run with mf=2, 3, 4 and 5. The value of $mf = 1$ is not considered because it favors the influence of outliers.

In addition it was done experiments using $\alpha = 0.05$ but they are not included here because experiments show that because the small sample size it turns in a really strict criterion, allowing in most of cases a single feature in the selected set which gives rise to a poor accuracy.

3.3.2 Design of experiments for IWSS relevance criteria.

The performance of different criteria is the Accuracy of the resulting models (i.e. running the classifier over the selected subset) is measured by using a 10cv. With respect to the classifier it is only considered Naive Bayes (NB), which is quite sensitive to the set of attributes used as input. Concretely it is used the WEKA [123] implementation of NB which models numerical variables by using uni-dimensional Gaussian distributions.

Datasets for experiments are 7 publicly obtained microarrays-based datasets, all of them related to cancer prediction. Datasets *Colon*, *Leukemia*, *Lymphoma* and *GCM* are the same used in [98] and can be downloaded in .arff format (e.g. WEKA data mining suite input format) from site `http://www.upo.es/eps/aguilar/datasets.html`. Datasets *DLBCL-Stanford*, *ProstateCancer* and *LungCancer-Harvard2* can be downloaded from site `http://sdmc.i2r.a-star.edu.sg/rp/`. Table 3.1 shows the number of features and records each dataset contains and also the accuracy achieved for each one when running a 10cv by using NB classifier. The last row shows the mean values for each column.

Table 3.1: Microarrays properties.

Dataset	**#Features**	**Size**	**Acc.(%)**
Colon	2000	62	53.23
Leukemia	7129	72	98.61
Lymphoma	4026	96	75.00
GCM	16063	190	66.84
DLBCL	4026	47	97.87
Prostate	12600	136	55.88
Lung	12533	181	98.34
Mean	8340	112	77.97

The design of the experiments is easy. IWSS is run by using each one of the proposed criteria (3 criteria $\times$ 4 α values = 12 criteria) over each one of the 7 datasets. In order to have a baseline results for the analysis of balance between number of selected features and obtained accuracy, IWSS is first run by using a *greedy* relevance criterion: $(\mathsf{AuxData} \triangleright \mathsf{BestData})$ is implemented as $(\mathsf{AuxData.av} > \mathsf{BestData.av})$ and the results of this algorithm called *SimpleBIRS* are shown in Table 3.2.

Table 3.2: Results for SimpleBIRS

Dataset	**#Features**	**Acc.(%)**
Colon	6.3	79.03
Leukemia	3.7	93.06
Lymphoma	11.7	77.08
DLBCL	3.7	91.49
Prostate	12.2	78.68
Lung	3.9	98.90
GCM	50.8	64.74
Mean	13.2	83.28

3.3.3 Results of experiments for relevance criteria in IWSS

Results of experiments are shown in Tables 3.3, 3.4 and 3.5.

Table 3.3: Results when considering t-test as relevance criterion.

Dataset	α =**0.1**		α =**0.15**		α =**0.2**		α =**0.25**	
	Acc.	**#f**	**Acc**	**#f**	**Acc.**	**#f**	**Acc.**	**#f**
Colon	82.26	2.4	77.42	2.6	77.42	3.1	82.26	3.4
Leukemia	86.11	1.6	86.11	1.6	90.28	2.2	90.28	2.2
Lymphoma	67.71	5.2	63.54	5.1	73.96	7.2	72.92	7.5
DLBCL	87.23	1.6	87.23	1.6	85.11	1.8	87.23	1.8
Prostate	74.26	4.1	75.74	4.7	75.00	6.8	75.00	6.6
Lung	96.13	1.6	96.13	1.6	97.24	2.4	97.24	2.4
GCM	54.21	12.4	60.53	13.3	59.47	19.4	60.53	18.5
Mean	78.27	4.1	78.10	4.4	79.78	6.1	**80.78**	6.1

From the tables, a straight-forward conclusion is that the more strict is the significance level (α or mf) used, the fewer the number of variables included in the selected subset. Although more observations of this type can be drawn, in order to back conclusions, a statistical analysis is carried out as described below.

Because of the multiple algorithms (criteria) and multiple datasets, statistical analysis is performed following the recommendations in [23]: Friedman test [37] followed

Table 3.4: Results when considering signed rank test as relevance criterion.

Dataset	$\alpha =$**0.1**		$\alpha =$**0.15**		$\alpha =$**0.2**		$\alpha =$**0.25**	
	Acc.	**#f**	**Acc**	**#f**	**Acc.**	**#f**	**Acc.**	**#f**
Colon	82.26	2.1	77.42	2.6	79.03	2.7	79.03	2.7
Leukemia	84.72	1.2	86.11	1.6	84.72	1.6	84.72	1.6
Lymphoma	69.79	3.9	63.54	5.2	64.58	5.3	64.58	5.3
DLBCL	80.85	1.3	87.23	1.6	87.23	1.5	87.23	1.5
Prostate	75.74	3.3	77.21	4.4	79.41	4.8	79.41	4.8
Lung	96.13	1.1	96.13	1.1	96.13	1.1	96.13	1.1
GCM	51.58	9.4	53.16	12.1	58.95	14.1	58.95	14.1
Mean	77.30	3.2	77.26	4.2	**78.58**	4.5	**78.58**	4.5

Table 3.5: Results when considering min folds better as relevance criterion.

Dataset	**mf=2**		**mf=3**		**mf=4**		**mf=5**	
	Acc.	**#f**	**Acc**	**#f**	**Acc.**	**#f**	**Acc.**	**#f**
Colon	80.65	3.8	80.65	3.0	83.87	2.2	74.19	1.9
Leukemia	87.50	2.5	86.11	1.7	83.33	1.2	83.33	1.1
Lymphoma	76.04	8.8	71.88	6.1	65.63	4.1	66.67	3.2
DLBCL	85.11	1.9	87.23	1.5	80.85	1.3	78.72	1.1
Prostate	77.94	11.1	79.41	7.2	75.74	3.7	75.74	2.6
Lung	97.24	2.7	96.13	1.7	96.69	1.2	96.13	1.0
GCM	64.21	36.6	64.74	24.5	50.53	11.4	47.37	5.8
Mean	**81.24**	9.6	80.88	6.5	76.66	3.6	74.59	2.4

by a post-hoc Holm test [42]. Friedman test is used for statistical comparison over three or more sets of values; in this case the inputs of the study are the set of mean accuracies (and mean number of features selected) computed for each microarray in each one of the 10 folds. The application of Friedman test only decides if there exists at least one set of values (e.g. one algorithm) which is different to at least another set of values (algorithm). Once this is known, then the post-hoc Holm test is run by choosing a control set of values and then comparing it with the rest of sets.

The comparison process is performed as follows:

1. First, it is identified for each *relevance* criterion (i.e. t-test, Wilcoxon and min better folds) the best significance (α) values. To do this, the Friedman test is run for each criterion regarding only accuracies as inputs and using also as input the results provided by SimpleBIRS (which is the baseline algorithm). So, in this step three Friedman tests are run (one per criterion) in order to identify those configurations ($\langle$criterion,significance-value$\rangle$) that are not statistically different from the accuracy achieved by SimpleBIRS. As in two of the three cases Friedman test returns that in fact there is at least one (statistically significant) different algorithm, the post-hoc Holm test is run by choosing as control the set of values provided by SimpleBIRS. Table 3.6 shows the results of this process, where crossed cells mean that they are significantly different from SimpleBIRS (the control) by using Holm test (p-value < 0.05). Thus, these algorithms are ruled out and therefore not considered in the subsequent steps.

Table 3.6: Results of step one: local comparison for each relevance criterion with respect to accuracy.

α=	**0.1**	**0.15**	**0.2**	**0.25**
mf=	**2**	**3**	**4**	**5**
t-test	~~78.3~~	~~78.1~~	~~79.8~~	80.8
Wilcoxon	77.3	77.3	78.6	78.6
MinValues	81.2	80.9	~~76.7~~	~~74.6~~
SimpleBIRS	● 83.3			

2. In the second step it is considered a single pool with all the survivor algorithms

from the previous phase. Thus, the previous process is repeated over the eight algorithms (1 using t-test criterion, 4 using Wilcoxon criterion, 2 using min folds better criterion and SimpleBIRS). Table 3.7 shows the results, where it can be seen how two new algorithms are ruled out when considering this global analysis.

Table 3.7: Results of step two: global comparison with respect to accuracy.

α=	**0.1**	**0.15**	**0.2**	**0.25**
mf=	**2**	**3**	**4**	**5**
t-test				80.8
Wilcoxon	~~77.3~~	~~77.3~~	78.6	78.6
MinValues	81.2	80.9		
SimpleBIRS	● 83.3			

3. To this point it has been obtained a set of six algorithms such that the global statistical analysis does not find significant differences among them. Therefore, it is time to consider the number of selected variables by them. Thus, the previous process (Friedman + Holm) is repeated but taking as inputs the set of values related to the mean number of selected features for each microarray in each one of the 10 folds. The results are shown in Table 3.8 (notice that now the control is the algorithm with the smallest selected subset). It can be observed that two configurations are ruled out, including the baseline algorithm.

Table 3.8: Results of step three: global comparison with respect to the number of selected features.

α=	**0.1**	**0.15**	**0.2**	**0.25**
mf=	**2**	**3**	**4**	**5**
t-test				6.1
Wilcoxon			● 4.5	4.5
MinValues	~~9.6~~	6.5		
SimpleBIRS	~~13.2~~			

Finally, from the originally 13 considered configurations (relevance criterion and strictness level), after the global statistical analysis, it has been obtained a set of four configurations whose results are *non-significantly different* neither with respect to accuracy nor with respect to the number of features. As it could be expected, neither maximum accuracy nor minimum number of selected features have been able to remain selected and only configurations with a good balance between accuracy and number of selected features have survived.

3.3.4 Conclusions on relevance criteria for IWSS

As pointed out in [98], incremental wrapper selection works better when using an objective relevance criterion (e.g. BIRS) than when using only an improvement in the mean accuracy as criterion (e.g. SimpleBIRS). However, in this study it has been detected that when using a t-test based relevance criteria, it is better to use a more relaxed confidence level (0.25 vs 0.1). The same conclusion about the strictness applies to the other two criteria (Wilcoxon and min folds better), therefore, it can be concluded that a more relaxed confidence level allows for the introduction of some extra feature which helps to improve the accuracy. Also of interest is to observe that using a non-parametric test is a clear alternative and even the use of a pure heuristic criterion which defends itself from noise and outliers by forcing to the selected subset to achieve an improvement in at least three (of the five) folds.

As a conclusion, it has been found that an optimal configuration for $\triangleright$ from Figure 3.1 is to impose that at least 2 or 3 values from the inner 5CV are greater when adding a new attribute than the respective folds without adding such attribute. Thus, this *MinFoldersBetter* with $mf = 2$ or $mf = 3$ will remain as the configuration used for IWSS in the following of this chapter.

3.4 Incremental wrapper-based selection with replacement

This section presents one of the main contributions of this chapter. The main advantage of IWSS algorithms is their linear complexity ($\mathcal{O}(n)$) in the number of wrapper evaluations. This is, of course, a favorable point when n grows large and exhaustive search ($\mathcal{O}(2^n)$) or even approximate search algorithms like SFS or SBS ($\mathcal{O}(n^2)$) are not feasible. However, there are of course disadvantages in the use of IWSS algorithms. Perhaps the most important one is due to its greedy behavior, that is, the algorithm always tries the best ranked features first and once a feature is included in the selected set, it is maintained therein until the end of the search.

What is proposed here is to alleviate some of these problems by allowing the algorithm not only to include a new feature in the selected set, but also to interchange it with one of the features previously included. Let's explain this idea by taking the Graphical Model (Bayesian network [83]) shown in figure 3.2 as starting point, but first it should briefly be explained how to read (in)dependence sentences from such graphical representation. BN's [52] have two components, a numerical part which encodes the probability distribution and the graphical component being a directed acyclic graph. From the graphical part we can codify (conditional) independences, so, the absence of a link between a pair of nodes is due to the conditional independence of such a pair of nodes, and the presence of a direct link between a pair of nodes indicates direct/strong relation or dependence between such a pair of nodes.

A Bayesian network (BN) [83] factorizes the joint probability distribution (P) by means of the product of the local probability functions associated to the nodes/variables (conditional probability of a variable given its parents in the graph). A BN is an I-MAP of P, therefore, if X is independent of Y given **Z** in the BN ($< X|\mathbf{Z}|Y >_{BN}$), then X is independent of Y given **Z** in P ($I(X|\mathbf{Z}|Y)_P$), that is, all independences stated by a BN hold in the joint probability distribution.

(In)Dependence sentences in a BN can be read by means of the *d-separation* criterion [83]. Let π be an undirected path (that is, without considering the direction of the arrows) from node U to V. Then, the path π between U and V is blocked by a set of nodes **Z** if an only if exactly one of the following holds:

1. π contains a sequential connection $X \rightarrow M \rightarrow Y$ and $M \in \mathbf{Z}$,
2. π contains a divergent connection $X \leftarrow M \rightarrow Y$ and $M \in \mathbf{Z}$, or
3. π contains a convergent connection $X \leftarrow M \rightarrow Y$ and neither M nor any of its descendants in the graph are included in **Z**.

If all the undirected paths between U and V are blocked by **Z**, then it is said that U and V are d-separated by **Z** in the BN (and consequently $I(U|\mathbf{Z}|V)_P$).

The Markov Blanket (MB) of a node X is formed by its parents ($pa(X) = \{Y : Y \rightarrow X\}$), its children ($ch(X) = \{Y : X \rightarrow Y\}$) and its spouses ($sp(X) = pa(ch(X)) - \{X\}$). By using the d-separation criterion it is easy to see that X is independent of the rest of the nodes (variables) given its MB: $< X|MB(X)|rest >_{BN}$.
Example 1. Let us consider the (graphical part of the) BN in figure 3.2. As we can observe it contains 10 variables $\{A, B, C = Class, D, E, I, J, K, L, M\}$ and can be matched with a classification problem where C is the class, $MB(C) = \{A, B, D, E\}$ are relevant for C and the rest of the variables ($\{I, J, K, L, M\}$) are irrelevant with respect to C if we know the value of variables in $MB(C)$.

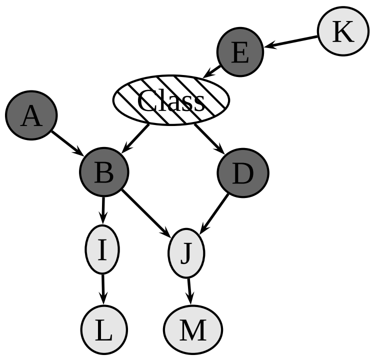

Figure 3.2: Network used in Example 1.

Therefore, in a feature selection problem $\{A, B, D, E\}$ should be the selected ones while $\{K, I, J, L, M\}$ should be discarded because they are irrelevant to the $class$ once we know the value of variables in $MB(C)$. Thus, ithe following behavior when using an IWSS algorithm should be expected :

- A *parent* or *child* variable has a strong (direct) dependence relation with the class and so it should be ranked in the first places by any information-based or correlation measure. Therefore, it should be considered for inclusion in the selected set before variables/features having a small degree of relation with the class.
- Things are different for *spouses*, because they are marginally independent of the class, that is, they are unrelated, and so they should be ranked after those variables that, to some degree, are relevant to the class. Spouses become relevant when the children common with the class or any of its descendants are included in the selected set, because then the path between the spouse and the class is no

longer blocked and so they are not d-separated. Since spouses should have worse ranking than children, when IWSS inspects spouses it is expected that common children are already included, and so spouses will also be included.

$\square$

However, the situation described in the previous example is the ideal one but not usually the actual one. That is, in a real problem we have to deal with (sometimes scarce) datasets, and most times that dataset is not faithful to a Bayesian network. As we know, this is the common situation specially when dealing with high-dimensional data sets, e.g. microarrays, where we have thousands of variables but only a few hundred instances. The following example illustrates a more realistic situation.

Example 2. In the case of having few records with respect to the number of variables, we cannot expect to deal with such accurate rankings. As an example, it is generated the quantitative part (conditional probabilities) for the BN in figure 3.2 and sampled a data set with 1000 instances for it. Then, the ranks are computed using SU for the dataset reduced to only the first 5, 10, 50 and 1000 records. The results are shown in Table 3.9.

Table 3.9: Rankings obtained/used in example 2

Rank Id	# records	SU-based Ranking
R1	5	E, J, D, K, L, I, A, B, M
R2	10	E, K, B, M, L, D, I, A, J
R3	50	E, D, J, B, K, A, I, L, M
R4	1000	E, D, B, K, I, J, A, L, M

As it can be seen, rank R4 reflects what we should expect in the ideal case, that is, parent and children $\{E, D, B\}$ being the first variables in the ranking. However, we should observe that 1000 records in the dataset means having almost all the possible configurations in the problem (2^{10}), which is highly unrealistic in real domains. Ranks R1, R2 and R3 show more realistic situations. We can observe also how the spouse A is always ranked in the last positions.

As E has been ranked as the first variable in all the cases, let us assume that it is always included in the selected set $\mathcal{S}$. Anyway, as this is not always the case, we should remark that the treatment of the class's parents and children is analogous.

- In rank R3 it can be assumed that the first three variables are directly included in $\mathcal{S}$ because at the time of analyzing them, they are not irrelevant (independent) to the class. Notice that E and D cannot be d-separated with respect to C because they are directly connected (e.g. dependent) while J is not d-separated from C given $\{E, D\}$ because the path $C \to B \to J$ is not blocked (condition 1 of d-separation). Thus, when IWSS analyzes B we have $\mathcal{S} = \{E, D, J\}$. Now, there are two possibilities:

  - The inclusion of B in $\mathcal{S}$ is judged as relevant by IWSS and so we get $\mathcal{S} = \{E, D, J, B\}$, which means that we have an extra feature in our selected set, because once B has been included in $\mathcal{S}$, variable J becomes irrelevant (d-separated) to $Class$.
  - The inclusion of B in $\mathcal{S}$ is judged as irrelevant by IWSS. That is, B is not d-separated from C given $\{E, D, J\}$ and although semantically it should be added, because of the previous inclusion of J and D, even if the accuracy obtained when including B can increase the current one, the improvement may not be big enough to be considered as relevant. Therefore, we get a selected set that, even though it is good, is of worse quality than the best possible one.

  These situations can be solved if instead of just analyzing the effect of adding a new attribute to $\mathcal{S}$ experiments are carried out also with the possibility of swapping each one of the already included variables with the new one, choosing in the end the best *relevant* option. In the example above we have:

$$\mathcal{S} = \{E, D, J\} \longrightarrow \begin{cases} \textsf{add}(B) & \to \quad \mathcal{S}_B = \{E, D, J, B\} \\ \textsf{swap}(E) & \to \quad \mathcal{S}_E = \{B, D, J\} \\ \textsf{swap}(D) & \to \quad \mathcal{S}_D = \{E, B, J\} \\ \textsf{swap}(J) & \to \quad \mathcal{S}_J = \{E, D, B\} \end{cases}$$

  and in this case $\mathcal{S}_J$ should be the best relevant subset. Notice that if none of the subsets tried is judged as relevant, then $\mathcal{S}$ remains unchanged.

- A similar situation to rank R3 appears in rank R2, where M is included before testing D (and J).

- R1 is the worst ranking with respect to the ideal one (R4) so in this case more problems were found when using IWSS. Thus, in ranking R1 it can be observed how J appears in the second place, while B is ranked at the end. Therefore, by following the ranking it can be expected to directly include $\{E, J, D\}$ and to reject K because E has been previously included. Now IWSS will also include L because the path $C \rightarrow B \rightarrow I \rightarrow L$ is not blocked; perhaps it includes I because the path $C \rightarrow B \rightarrow I$ is not blocked and I should have more impact on $Class$ than L; A[1] should also be included; and perhaps B for the same reason as I. That is, the execution of IWSS could obtain $\mathcal{S} = \{E, J, D, L, I, A, B\}$ or $\mathcal{S} = \{E, J, D, L, A, B\}$ or $\mathcal{S} = \{E, J, D, L, I, A\}$ or $\mathcal{S} = \{E, J, D, L, A\}$ depending on the judgment of relevance performance when trying to include I and B. Obviously, none of these sets is the desired one.

Let us analyze what might happen when starting at the point where $\mathcal{S}$ = $\{E$, J, D, L $\}$ and using replacement:

– Testing I. In this case the best choice should be swap(I,L) so $\mathcal{S}$ = $\{E$, J, D, $I\}$.
– Testing A. In this case the best choice should be add(A) so $\mathcal{S}$ = $\{E$, J, D, I, $A\}$.
– Testing B. In this case the best choice should be swap(B,I) or swap(B,J) so $\mathcal{S} = \{E, J, D, B, A\}$ or $\mathcal{S} = \{E, B, D, I, A\}$.
– Testing M. In this case all the operations should be judged as non relevant and so $\mathcal{S}$ is not updated.

Therefore, a better subset than the ones listed above is obtained, because even there is a redundant variable included in the final set, there is less redundancy and the desired subset is fully contained.

□

IWSS with replacement, named as IWSS$_r$, i.e., IWSS incorporating the proposal to obtain better selected subsets is shown in Figure 3.3, where the novelties with respect to IWSS start at line 13.

Of course, having to test swapping and not only addition when analyzing each variable represents an increment in the complexity of the algorithm. In the worst case,

[1] Notice that there is no problem with the inclusion of *spouse* A, even if it is ranked after B, because some descendants of B appear earlier in the ranking.

that is, if the add operation is selected at each step, then we get $\mathcal{O}(n^2)$ (as in sequential forward selection). However, that means that there is no selection at all and the n variables are included in $\mathcal{S}$, which is not usual at all. In practice, the complexity is $\mathcal{O}(mn)$, m being the number of selected variables. In large datasets (e.g. microarrays) $m << n$ and so we are quite far from $\mathcal{O}(n^2)$. More will be discussed about complexity in Section 3.6.5.

```
In    $\mathbf{T}$ training, $M$ measure, $\mathcal{C}$ classifier
Out   $\mathcal{S}$
1     list $R$ = {}      // The ranking
2     for each attribute $A_i \in \mathbf{T}$
3       Score=$M_{\mathbf{T}}(A_i$, class)
4       insert $A_i$ in $R$ according to Score
5     $\mathcal{S}$ = $\{R[1]\}$
6     BestData = evaluate($\mathcal{C}, \mathcal{S}_{aux}, \mathbf{T}$)
7     for i = 2 to $n$      // $n = R.size()$
8       bestOp = null;
9       for j = 1 to M      // M = $\mathcal{S}$.size()
10        $\mathcal{S}_{aux}$ = update($\mathcal{S}$,swap($\mathcal{S}[j], R[i]$))
11        AuxData = evaluate($\mathcal{C}, \mathcal{S}_{aux}, \mathbf{T}$)
12        if (AuxData $\triangleright$ BestData)
13          bestOp = update($\mathcal{S}, swap(\mathcal{S}[j], R[i])$)
14          BestData = AuxData
15      $\mathcal{S}_{aux}$ = $\mathcal{S} \cup \{R[i]\}$
16      AuxData = evaluate($\mathcal{C}, \mathcal{S}_{aux}, \mathbf{T}$)
17      if (AuxData $\triangleright$ BestData)
18        bestOp = add(R[i])
19        BestData = AuxData
20      if (bestOp $\neq$ null)
21        update($\mathcal{S}$,bestOp)
```

Figure 3.3: Algorithm: IWSS with replacement.

3.5 IWSS with early stopping

One of the characteristics of IWSS is that it runs over the whole ranking, giving each attribute in the data set the chance of being selected. This has the disadvantage of increasing the number of wrapper evaluations carried out, specially when replacement is used. At this point, we can ask ourselves: is it really necessary to run over the whole ranking? From the description/discussion in the previous section and because the score used to build the rank only measures direct (marginal) relevance relations of one attribute with the class, we can argue that all the relevant attributes will be ranked in the first part of the rank, except *spouses*. Thus, from a semantic point of view (e.g. it is desirable to understand the interrelations among the class and the variables relevant to it), it would be nice to identify all the relevant variables, but for a *purely predictive* task such as classification, perhaps it is not necessary to include all these variables.

From extensive experiments with different versions of IWSS, it was in fact observed that variables with a bad ranking are almost never included in the selected subset $\mathcal{S}$. A possible explanation of this fact is that although these variables have an indirect relevance relation with the class, once the variables having a stronger relevance relation have been included in $\mathcal{S}$, their contribution to the classification task may not be important enough to be judged as relevant. Furthermore, the utility of such types of variables depends on the complexity of the classifier used. For example, if Naive Bayes is used, then adding spouses makes no sense, because they are marginally independent with respect to the class, and so $P(X = x_i|C)$ would be (almost) the same for all states x_i of spouse X. That is, we are adding an irrelevant variable.

Because of these reasons, and with the aim of saving some wrapper evaluations and so speeding up the IWSS process, the *early stopping* criterion proposes to stop the main cycle before arriving at the end of the ranking. Of course, the effect of early stopping should be more beneficial for IWSS$_r$ than for IWSS. In the following, implementations of IWSS and IWSS$_r$ with early stopping are denoted as IWSS$_s$ and IWSS$_{r,s}$ respectively.

The decision now is when the main cycle has to be stopped. The proposal here is to use a threshold, $\theta \in (0, 1]$, which will limit the percentage of attributes to be studied, that is, considered for inclusion in $\mathcal{S}$. However, instead of using this threshold as a crisp limit, we prefer to use it in an adaptive way, that is, each time a new feature X enters $\mathcal{S}$ the number of features to be visited is updated.

In order to incorporate early stopping, IWSS and IWSS$_r$ must be modified by adding the following sentences:

- At the initialization: $l = \theta \cdot n$
- The main cycle is replaced by: for i = 2 to l do
- As the last sentence of the main cycle we include:

```
    if ($\mathcal{S}$ has been modified)
     $l = i + \theta \cdot (n - i)$
```

For example, if we have $\theta = 0.1$ and $n = 1000$, then initially, we set $l = 100$ and so it is expected to analyze only the first 100 variables in the ranking. Then, if the first variable introduced in $\mathcal{S}$ is in position 50 of the ranking, then l is updated as $50 + 0.1 \cdot (1000 - 50) = 145$, and so the number of variables expected to visit is enlarged up to the first 145 variables in the ranking.

3.6 Experiments with Replacement and Early Stopping

This section performs a series of experiments with the aim of analyzing the type of improvement introduced by the last two proposals explained in this chapter: replacement and early stopping. To do this, a set of 12 high-dimensional datasets is used (a superset of those used in [98]) ranging between 500 and 100000 predictive attributes. The idea is first to analyze the behavior of IWSS when using replacement, and then to study how early stopping affects the quality and the efficiency of IWSS algorithms.

3.6.1 Datasets

Experiments will be run over 12 publicly obtained datasets. Seven of them are the microarrays related to cancer prediction introduced in Section 3.3.2. In addition to these microarrays datasets, another five datasets will be used which are known for being provided in the NIPS 2003 feature selection challenge (Arcene, Madelon, Dorothea, Dexter, and Gisette). Table 3.10 shows the number of features and records/instances each dataset contains. As it can be seen, these datasets are characterized by a large number of predictive attributes but a small number of available instances, this trend being sharper in the case of microarray datasets.

Table 3.10: Microarrays properties.

Dataset	**#Features**	**#Instances**
Colon	2000	62
Leukemia	7129	72
Lymphoma	4026	96
DLBCL	4026	47
Prostate	12600	136
Lung	12533	181
GCM	16063	190
Arcene	10000	100
Madelon	500	2000
Dorothea	100000	800
Dexter	20000	300
Gisette	5000	6000
Mean	16156	2011

3.6.2 Classifiers and baseline algorithms

For the experiments three standard classifiers belonging to three different paradigms are considered: Naive Bayes, KNN (K=1) and C4.5 (Section 2.3), using the implementation of these algorithms included in the WEKA data mining suite [123] and in all the cases default parameter setting is applied. Of these three classification algorithms, C4.5 implements its own attribute selection process, while Naive Bayes (NB) is known to be quite sensitive to noise and redundant attributes, so special attention to this classifier is payed.

With respect to feature selection algorithms, and in order to test the two proposals, the following three baseline algorithms are also used:

- Fast Correlation-based Filter (FCBF) algorithm [128]. This algorithms lies in the filter approach and uses a correlation measure in order to remove redundant attributes while retaining the relevant ones.

- Sequential Forward Selection (SFS) algorithm [41]. This algorithm is a greedy one that first includes the best attribute found, then looks for the second-best by trying to add any possible attribute to the one previously selected, and continues in this way until no improvement is obtained when adding a new attribute. SFS is usually combined with wrapper evaluation and so improvement is measured in terms of accuracy. In the worst case it carries out $\mathcal{O}(n^2)$ wrapper evaluations. In practice, if m attributes are finally selected then it carries out about $(m+1) \cdot n$ wrapper evaluations ($n + (n-1) + ... + (n-m)$).

- Incremental Wrapper-based Subset Selection (IWSS) algorithm is the approach to improve by using replacement. Therefore, it is used as baseline, using the version that implements MinFoldersBetter as relevance criterion (see Section 3.3).

As validation criterion, standard 10-fold cross-validation is run. Therefore, the results reported in this section are the average of the results obtained for each one of the folds.

3.6.3 Testing IWSS with replacement

First, the impact of including replacement on incremental selection is tested. Table 3.11 shows the results when considering $mf = 2$ and $mf = 3$ as relevance criterion. A superscript is used with the value of mf and r as subscript when using replacement. Therefore, initially four algorithms are tested: IWSS$_r^2$, IWSS$_r^3$, IWSS^2 and IWSS^3.

From the results, it is observed how FCBF always includes (many) more attributes in the selected subset than the wrapper-based ones. With respect to accuracy, there are some differences depending on the dataset, but when carrying out a statistical analysis with multiple datasets and multiple algorithms, no significant difference is observed in most of the cases (see Section 3.6.6). Returning to the number of selected attributes, IWSS selects less attributes than SFS in all the datasets except GCM. In this case it is also worth commenting that IWSS is faster than SFS because it only carries out n wrapper evaluations.

When *replacement* is introduced, then the number of variables selected by IWSS$_r$ is smaller than that by SFS and IWSS. With respect to complexity, as it will analyzed in Section 3.6.5, IWSS$_r$ carries out almost the same number of wrapper evaluations as SFS (but it selects fewer attributes). Of course, we can also note how the relevance criterion also plays its role, thus versions requiring being better in 3 folders to be judged as relevant, are more restrictive than versions that require being better in only 2 folders, therefore IWSS^3 and IWSS$_r^3$ selects less attributes than IWSS^2 and IWSS$_r^2$.

With respect to the classifier used, a similar behavior is detected in the three cases.

Table 3.11: Results of SFS, FCBF, IWSS and IWSS$_r$ for classifiers NB, kNN and C4.5.

DataSet	**SFS**		**FCBF**		**IWSS^2**		**IWSS^3**		**IWSS$_r^2$**		**IWSS$_r^3$**	
Colon	83.87	5.9	77.42	14.6	80.65	3.8	80.65	3	83.87	2.8	82.26	2.1
Leukemia	87.50	3.2	95.83	45.8	87.50	2.5	86.11	1.7	87.50	2	86.11	1.6
Lymphoma	83.33	7.1	78.13	291	76.04	8.8	71.88	6.1	80.21	5.9	73.96	5.6
DLBCL	80.85	3.6	97.87	49.9	85.11	1.9	87.23	1.5	80.85	1.8	87.23	1.5
Prostate	75.00	5.4	61.03	35.8	77.94	11.1	79.41	7.2	78.68	7	79.41	5
Lung	93.92	2.5	99.45	115.2	97.24	2.7	96.13	1.7	97.24	2.4	95.58	1.6
GCM	58.42	18.3	68.95	57.1	64.21	36.6	64.74	24.5	59.47	19.9	58.42	14.5
Arcen	68.00	4.6	70.00	34.2	70.00	13.4	70.00	6.3	72.00	6.2	71.00	5.5
Madelon	60.75	6.5	61.75	4.6	59.85	13.3	60.00	8.9	60.50	8.0	59.90	6.1
Dorothea	91.25	13.2	92.63	92.8	93.50	7.4	93.63	3.6	92.88	6.3	92.63	3.7
Dexter	76.00	13.8	86.00	34.3	81.00	19.6	81.67	12.9	83.00	12.9	80.00	10.0
Gisette	94.05	26.09	88.03	30.4	94.68	112.6	94.28	73.4	94.07	30.7	93.28	20.8
Mean	**79.41**	**9.2**	**81.42**	**67.1**	**80.64**	**19.5**	**80.48**	**12.6**	**80.86**	**8.8**	**79.98**	**6.5**

(a) NB

DataSet	**SFS**		**FCBF**		**IWSS^2**		**IWSS^3**		**IWSS$_r^2$**		**IWSS$_r^3$**	
Colon	66.13	4.8	80.65	14.6	82.26	6.3	75.81	4.6	77.42	4.9	82.26	3.6
Leukemia	88.89	2.3	94.44	45.8	88.89	2.8	86.11	2.1	87.5	2.2	90.28	1.9
Lymphoma	79.17	8.2	91.67	291.0	81.25	12.5	78.13	8.2	78.13	7.7	78.13	6
DLBCL	89.36	3.4	95.74	49.9	85.11	3.5	78.72	2.1	85.11	2.5	78.72	2.1
Prostate	78.68	4.6	77.94	35.8	86.03	8.6	86.76	5.9	88.97	5.3	89.71	4.3
Lung	95.58	2.6	99.45	115.2	96.13	2.7	95.58	2.2	96.69	2.5	95.03	2
GCM	56.84	17.9	63.16	57.1	65.26	34.1	51.05	2.5	55.79	20.5	57.89	15.4
Arcene	71.00	4.6	66.00	34.2	76.00	13.2	71.00	7.9	72.00	6.9	77.00	5.2
Madelon	52.35	6.5	55.50	4.6	88.00	11.7	87.85	10.6	87.85	8.1	88.70	8.4
Dorothea	91.50	13.2	91.88	92.8	91.88	18.2	92.00	8.7	93.13	15.9	91.63	8.1
Dexter	73.67	13.8	80.00	34.3	83.33	24.6	80.00	15.5	80.21	20.1	76.67	12.9
Gisette	92.15	26.9	90.03	30.4	95.97	100.4	96.05	71.6	-		-	
Mean	**77.94**	**9.1**	**82.21**	**67.1**	**85.01**	**19.9**	**81.59**	**11.8**	**82.07**	**8.8**	**82.37**	**6.4**

(b) KNN

DataSet	**SFS**		**FCBF**		**IWSS^2**		**IWSS^3**		**IWSS$_r^2$**		**IWSS$_r^3$**	
Colon	80.65	3.3	85.48	14.6	79.03	2.7	80.65	2	79.03	2.3	82.26	2
Leukemia	87.50	1.6	81.94	45.8	83.33	1.1	83.33	1	84.72	1.1	84.72	1
Lymphoma	71.88	8.2	82.29	291.0	75.00	8.9	78.13	6.2	77.08	6.4	77.08	5.1
DLBCL	78.72	1.5	72.34	49.9	76.60	1.5	76.60	1.2	82.98	1.3	76.60	1.2
Prostate	78.68	6.9	79.41	35.8	88.24	4.9	88.97	3.8	79.41	4.5	88.24	3.5
Lung	93.92	1.8	96.13	115.2	95.03	1.3	95.03	1.3	93.92	1.3	95.03	1.3
GCM	41.05	14.8	51.58	57.1	45.26	23.4	46.32	16	41.58	15.6	40.53	11.3
Arcen	65.00	4.6	85.00	34.20	82.00	7.5	73.00	5.2	80.00	5.4	77.00	4.5
Madelon	61.55	6.5	61.90	4.60	77.25	23.1	77.90	19	77.90	13.4	78.10	12.0
Dorothea	91.25	13.2	89.88	92.80	91.63	9.7	92.00	7.7	91.88	8.5	97.75	7.5
Dexter	79.67	13.8	81.67	34.30	81.33	13.2	81.33	7.7	80.67	9.6	81.33	7.2
Gisette	93.73	26.9	91.32	30.40	93.70	67.8	93.63	45.4	93.75	35.8	93.28	27.6
Mean	**76.97**	**8.6**	**79.91**	**67.1**	**80.70**	**13.8**	**80.57**	**9.7**	**80.24**	**8.8**	**80.99**	**7.0**

(c) C4.5

Therefore, for the sake of simplicity, from now on the study shall continue using only Naive Bayes.

3.6.4 Testing early stopping

In this section, the first performed is early stopping in combination with replacement, giving rise to algorithm IWSS$_{r,s}$, but later, for the sake of completeness, it is also analyzed the behavior of using early stopping without replacement (IWSS$_s$). In the analysis three different values have been tested for θ: 0.2, 0.4 and 0.6. The results obtained by these algorithms are shown in Table 3.12.

From the results it is observed how the number of selected attributes *slightly* increases with the value of θ. However, the difference both in number of attributes and in accuracy is not statistically significant (as discussed in Section 3.6.6). Of course, there is a difference in the efficiency (CPU time) of the algorithms.

3.6.5 Testing Complexity

To analyze the complexity of the algorithms, focus is on the number of wrapper evaluations carried out. Obviously, no data is reported for basic IWSS because it has $\mathcal{O}(n)$ complexity and, in fact, exactly n wrapper evaluations are carried out. Furthermore, because no significant difference arises with respect to θ, the smallest tested value ($\theta = 0.2$) is selected for the following analysis. Table 3.13.(a) shows the number of wrapper evaluations carried out by the different algorithms. With respect to the complexity order, we know that in the worst case SFS and all the variants of IWSS$_r$ have worst-case complexity $\mathcal{O}(n^2)$. However, in practice the algorithms are far from this number of evaluations, so Table 3.13.(b) shows the minimum and maximum complexity order computed from the actual number of evaluations carried out over the twelve datasets, and also the fitted value which minimizes the root mean squared error.

From the results, is is observed how SFS is a bit more complex than IWSS$_r$ and also how the *estimated* actual complexity order for both algorithms is far from n^2. On the other hand, when early stopping is introduced the analysis is clear: the number of evaluations carried out by IWSS with replacement decreases to be linear, while when replacement is not used the complexity order decreases to be sub-linear.

Table 3.12: Results obtained for NB when using early stopping.

	IWSS$^2_{r,s}$						**IWSS$^3_{r,s}$**					
	θ=**0.2**		θ=**0.4**		θ=**0.6**		θ=**0.2**		θ=**0.4**		θ=**0.6**	
Dataset	**Acc.**	**#f**	**Acc.**	**#f**	**Acc**	**#f**	**Acc**	**#f**	**Acc.**	**#f**	**Acc.**	**#f**
Colon	83.9	2.5	83.9	2.5	83.9	2.5	82.3	2.1	82.3	2.1	82.3	2.1
Leukemia	87.5	2.0	87.5	2.0	87.5	2.0	86.1	1.6	86.1	1.6	86.1	1.6
Lymphoma	80.2	5.9	80.2	5.9	80.2	5.9	77.1	5.5	77.1	5.5	74.0	5.6
DLBCL	80.9	1.7	80.9	1.7	80.9	1.7	87.2	1.5	87.2	1.5	87.2	1.5
Prostate	77.9	5.6	78.7	5.9	77.9	6.6	80.9	4.7	79.4	4.9	78.7	4.9
Lung	97.2	2.4	97.2	2.4	97.2	2.4	95.6	1.6	95.6	1.6	95.6	1.6
GCM	58.9	17.5	61.1	18.1	58.4	18.8	60.0	13.0	59.5	13.7	59.5	13.7
Arcene	72.0	5.2	69.0	5.6	72.0	5.9	73.0	4.3	72.0	5.0	70.0	5.1
Madelon	60.4	5.9	60.4	6.7	59.9	7.3	60.2	4.1	60.4	5.4	60.0	5.6
Dorothea	93.0	5.2	92.9	6.3	92.9	6.3	92.6	3.5	92.6	3.7	92.6	3.7
Dexter	83.7	10.8	83.7	11.4	83.0	11.8	80.7	9.2	81.0	9.3	81.0	9.8
Gisette	93.9	25.3	93.9	28.6	94.1	30.2	93.1	18.5	93.1	19.4	93.2	20.1
Mean	**80.8**	**7.5**	**80.8**	**8.1**	**80.7**	**8.5**	**80.7**	**5.8**	**80.5**	**6.1**	**80.0**	**6.3**

(a) With replacement

	IWSS$^2_{s}$						**IWSS$^3_{s}$**					
	θ=**0.2**		θ=**0.4**		θ=**0.6**		θ=**0.2**		θ=**0.4**		θ=**0.6**	
Dataset	**Acc.**	**#f**	**Acc.**	**#f**	**Acc.**	**#f**	**Acc**	**#f**	**Acc**	**#f**	**Acc.**	**#f**
Colon	80.7	3.5	80.7	3.5	80.7	3.6	80.7	3.0	80.7	3.0	80.7	3.0
Leukemia	87.5	2.5	87.5	2.5	87.5	2.5	87.2	1.5	86.1	1.7	86.1	1.7
Lymphoma	76.0	8.5	76.0	8.7	76.0	8.7	86.1	1.7	71.9	6.1	71.9	6.1
DLBCL	85.1	1.8	85.1	1.9	85.1	1.9	71.9	6.1	87.2	1.5	87.2	1.5
Prostate	78.7	10.8	78.7	10.8	78.7	11.0	77.9	6.8	79.4	6.9	79.4	6.9
Lung	97.2	2.7	97.2	2.7	97.2	2.7	96.1	1.7	96.1	1.7	96.1	1.7
GCM	63.7	33.7	64.7	35.4	65.3	35.9	57.9	20.8	61.6	22.9	62.1	23.2
Arcene	72.0	12.1	70.0	13.2	70.0	13.4	70.0	5.7	69.0	5.9	69.0	6.1
Madelon	60.3	11.7	59.8	13.1	59.8	13.2	60.2	7.2	59.9	8.5	60.1	8.8
Dorothea	93.5	7.4	93.5	7.4	93.5	7.4	93.6	3.5	93.6	3.6	93.6	3.6
Dexter	83.0	17.2	83.3	18.5	81.3	19.2	80.7	11.6	81.0	12.1	81.3	12.8
Gisette	94.8	109.3	94.7	112.4	94.7	112.6	94.3	72.0	94.3	73.1	94.3	73.4
Mean	**81.0**	**18.4**	**80.9**	**19.2**	**80.8**	**19.3**	**79.7**	**11.8**	**80.1**	**12.3**	**80.2**	**12.4**

(b) Without replacement

Table 3.13: Number of evaluations and estimated complexity order ($\theta = 0.2$ is used for algorithms implementing early stopping)

Dataset	SFS	IWSS^2_r	IWSS^3_r	IWSS$^2_{r,s}$	IWSS$^3_{r,s}$	IWSS^2_s	IWSS^3_s
Colon	13800	7277	6181	1177	1025	627	621
Leukemia	29942	21378	18489	2153	1942	1441	1472
Lymphoma	32611	27663	26238	3169	3376	982	962
DLBC	18520	11134	10062	1094	1009	815	807
Prostate	80640	94508	73422	12196	9370	4494	3949
Lung	43866	42604	32570	4274	5544	2516	3525
GCM	310016	309750	231983	50754	44233	9321	8987
Arcene	56000	67359	58928	8202	7149	3928	3183
Madelon	3750	3818	3078	700	397	287	249
Dorothea	1420000	441346	249337	73246	45704	28303	25667
Dexter	296000	255027	211480	32536	29420	6423	5354
Gisette	184500	137051	96267	32340	23999	2526	2408
Mean	**207470**	**118243**	**84836**	**18487**	**14431**	**5139**	**4765**

(a) Number of wrapper evaluations carried out

	SFS	IWSS^2_r	IWSS^3_r	IWSS$^2_{r,s}$	IWSS$^3_{r,s}$	IWSS^2_s	IWSS^3_s
min	1.13	1.12	1.08	0.84	0.83	0.81	0.81
fitted	1.23	1.13	1.08	0.97	0.93	0.89	0.88
max	1.42	1.39	1.35	1.22	1.18	0.94	0.94

(b) Estimated complexity order ($\mathcal{O}(n^x)$)

3.6.6 Statistical Analysis

In order to corroborate conclusions, a statistical analysis is run based on the use of the Friedman test [37] followed by a post-hoc Holm test [42], as recommended in [23] when having to analyze multiple algorithms.

Tests for the case of $mf = 2$ and $mf = 3$ are performed separately, so for each of these two cases the set of tested algorithms contains: SFS, FCBF, IWSS, IWSS$_r$, IWSS$_s$ (θ=0.2,0.4,0.6) and IWSS$_{r,s}$ (θ=0.2,0.4,0.6). Since these algorithms could not be run using classifier c4.5 nor classier k-NN for database *Gisette* due to its high computational load (see Table 3.13), this database is not included in the tests (and values in Table 3.14 are the mean of the other 11 datasets). Thus, the input for the Friedman test for each classifier consists of 10 algorithms with 11 input values (mean of the 10CV for each of the 11 datasets). The procedure of analysis is similar to the one used in previous section; that is, for each classifier: first, run the Friedman test for accuracy values. If any statistical difference is found, then the Holm test is performed to find differences with the best algorithm found (control algorithm). Secondly, attention is payed to the number of selected attributes, repeating the previous process but considering only those algorithms not found to be statistically different with respect to accuracy. Finally, the remaining algorithms are regarded as not being statistically different.

In order to get more general conclusions, the analysis is performed over the three classifiers considered in Section 3.6.2.

Table 3.14 shows the results for each classifier when running tests based on accuracy, with $mf = 2$ (part (a)) and $mf = 3$ (part (b)) respectively. For both $mf = 2$ and $mf = 3$, the Friedman test does not find any algorithm to be statistically different from the others.

Then, Table 3.14.(c) and (d) shows the result for the analysis over the number of selected variables. In this case, the lower this number is, the better the algorithm performed. As it can observed, algorithm IWSS with replacement and early stopping ($\theta = 0.2$) is always chosen as control classifier (marked with a $\bullet$ in the table); then, a Holm test is run and the algorithms found to be statistically worse than the control have their cells marked grey in the table. Thus, algorithm IWSS$_{sr}$ ($\theta = 0.2$) is found to be statistically better than all the others algorithms except for the other three algorithms also performing replacement and/or early stopping. This proves that the proposed improvements for canonical IWSS (replacement and early stopping) achieve statistically lower cardinality in the final subset of selected features without downgrading accuracy performance.

3.7 Optimizing IWSS with replacement for Naive Bayes by embedding the classifier

In previous Sections it has been shown how IWSS$_r$ has the same worst-case complexity as sequential forward selection $\mathcal{O}(n^2)$, but that in practice the algorithm is far from this worst case. Furthermore, with the use of early stopping the number of evaluations is drastically reduced (without degrading its performance), achieving a linear or sub-linear number of wrapper evaluations. This Section goes one step further and shows how for those classifiers that allow incremental construction with respect to included variables, the efficiency of the selection algorithm can be significantly improved.

Up to now we have dealt with the wrapper algorithm by managing the classifier

Table 3.14: Results of the statistical analysis

	SFS	**FCBF**	**IWSS**	**IWSS**$_s$			**IWSS**$_r$	**IWSS**$_{r,s}$		
				θ =0.2	θ =0.4	θ =0.6		θ =0.2	θ =0.4	θ =0.6
NBayes	78.08	80.82	79.37	79.89	79.64	79.39	79.65	79.60	79.43	79.60
c4.5	75.44	78.88	79.52	79.43	79.51	79.51	78.92	79.08	79.04	79.01
k-NN	76.65	81.49	84.01	83.25	83.54	83.69	82.05	81.86	82.42	82.37

(a) Statistical tests regarding accuracy (MinFolders=2)

	SFS	**FCBF**	**IWSS**	**IWSS**$_s$			**IWSS**$_r$	**IWSS**$_{sr}$		
				θ =0.2	θ =0.4	θ =0.6		θ =0.2	θ =0.4	θ =0.6
NBayes	78.08	80.82	79.22	78.63	79.06	79.10	78.77	79.44	79.38	78.81
c4.5	75.44	78.88	79.39	79.11	79.33	79.39	79.88	79.36	79.31	79.44
k-NN	76.65	81.49	80.27	80.56	80.69	80.25	82.36	81.38	82.08	82.22

(b) Statistical tests regarding accuracy (MinFolders=3)

	SFS	**FCBF**	**IWSS**	**IWSS**$_s$			**IWSS**$_r$	**IWSS**$_{sr}$		
				θ =0.2	θ =0.4	θ =0.6		θ =0.2	θ =0.4	θ =0.6
NBayes	~~7.6~~	~~70.5~~	~~11.0~~	~~10.3~~	~~10.8~~	~~11.0~~	6.8	●5.8	6.3	6.5
c4.5	~~6.93~~	~~70.5~~	~~8.85~~	8.1	~~8.7~~	~~8.7~~	6.7	●5.3	5.9	6.1
k-NN	7.4	~~70.5~~	~~12.6~~	~~11.9~~	~~12.4~~	~~12.5~~	8.8	●7.9	8.3	8.6

(c) Statistical tests regarding number of selected subsets (MinFolders=2)

	SFS	**FCBF**	**IWSS**	**IWSS**$_s$			**IWSS**$_r$	**IWSS**$_{sr}$		
				0.2	0.4	0.6		0.2	0.4	0.6
NBayes	~~7.6~~	~~70.5~~	~~7.0~~	~~6.4~~	~~6.8~~	~~7.0~~	5.2	●4.6	4.9	5.0
c4.5	~~6.93~~	~~70.5~~	~~6.46~~	5.8	~~6.3~~	~~6.4~~	5.1	●4.3	4.6	4.9
k-NN	~~7.4~~	~~70.5~~	~~8.2~~	~~7.5~~	~~8.0~~	~~8.2~~	6.4	●5.6	5.9	6.1

(d) Statistical tests regarding number of selected subsets (MinFolders=3) like a *black box*. That is, each time we need to evaluate a new subset $\mathcal{S}$, then a new classifier is constructed from scratch by using the projection of the dataset over the candidate subset, $\mathbf{T}^{\downarrow(\mathcal{S}\cup C)}$, and validated by classifying each instance of the test set. In this case, and due to the 5-fold cross validation used to assess the goodness of a candidate subset, that means learning five classifiers (using $\frac{4}{5}$ of the instances each time) and validating each one over $\frac{1}{5}$ of the instances in the training set.

For example, if NB is the classifier used, which has $\mathcal{O}(tn)$ at training[1] and $\mathcal{O}(cn)$ for classifying one instance, then we get the following complexity for IWSS and IWSS$_r$ respectively:

- IWSS.
  - Learning: $5 \cdot \mathcal{O}(\frac{4}{5}tm) \cdot \mathcal{O}(n)$, that is, $4 \cdot \mathcal{O}(tmn)$ or simply $\mathcal{O}(tmn)$ by removing the constant. In the worst case $m = n$ and so we get $\mathcal{O}(tn^2)$.
  - Classification[2]: $5 \cdot \frac{t}{5} \cdot \mathcal{O}(cm) \cdot \mathcal{O}(n)$, that is, $\mathcal{O}(tcmn)$. In the worst case ($m = n$) and so we get $\mathcal{O}(tcn^2)$.
- IWSS$_r$. The main difference is that now each time a variable is tested to be included or not in the selected subset, $m + 1$ candidate subsets are evaluated (adding the new one and trying the m possible replacements). Applying the previous computations we get:
  - Learning: $\mathcal{O}(tm^2n + tmn)$. And in the worst case we have $\mathcal{O}(tn^3 + tn^2)$.
  - Classification: $\mathcal{O}(cm^2tn + cmtn)$. And in the worst case we have $\mathcal{O}(cn^3t + ctn^2)$.

If the classifier is used as a *black box*, this is what we get, but for some classifiers that can be constructed incrementally with respect to the number of variables included, we can do it better by embedding the process of learning and validating the classifier inside the wrapper process. This is, for example, the case of Naive Bayes, which is described below, distinguishing between learning and classification phases.

[1] Notation: n is the number of attributes in the training set, t is the number of instances in the training set, m is the number of attributes selected by the FSS algorithm and c the number of classes ($c = |C|$).

[2] In this analysis by classification we do not refer to the cost of classifying an instance with the resulting classifier, but to the classification process of the test folders carried out in the inner validation processes.

Learning.-

Because of its independence assumption, no structural learning is required in NB, so only parameter learning is required. Thus, in IWSS and IWSS$_r$ when a new attribute is studied, learning is reduced to estimating its conditional probability given the class. Therefore, avoiding the constant (4), time complexity is $\mathcal{O}(t)$ in each call, yielding $\mathcal{O}(tn)$ in the whole process. Notice that this complexity order is the same in IWSS and IWSS$_r$ because the conditional probability table for each attribute is computed only once. With respect to space complexity, we simultaneously need to store the probability tables of the attributes in the currently selected subset plus the one for the attribute under study, that is, $\mathcal{O}(mcr)$, r being the max number of values for a predictive attribute. Therefore, both in time and space, the incremental processes have the same complexity as if we were learning the whole model one time from scratch.

Classification.-

In order to deal with classification in an incremental way, the first step is to move computations to a log scale. In this way we can take into account the impact of a new variable by addition or discard it by subtraction (this prevents us from having problems with zero values). Thus, instead of computing

$$p(x_1, x_2, \ldots, x_m|c) = p(c) \cdot \prod_{i=1}^{m} p(x_i|c)$$

now, we compute:

$$Lp(x_1, x_2, \ldots, x_m|c) = log(p(c)) + \sum_{i=1}^{m} log(p(x_i|c))$$

The second step is to maintain a table in memory with c rows for each instance (ordered by using the test folder of the 5-fold cross validation). Columns of this table store $Lp(x_i|c)$ for each attribute in the current selected subset, plus a column containing the value $Lp(x_1, \ldots, x_m|c)$ for the instance projected over the current selected subset. Table 3.15 shows the structure of the table, where $q = t/5$ and $i_{11}^{\downarrow X_1}$ denotes the projection of the first instance of the first folder over variable X_1, that is, we take the value of variable X_1 in such an instance.

By using the previous table, IWSS and IWSS$_r$ perform incremental classification as follows:

- IWSS. Let us assume that $S = \{X_1, \ldots, X_m\}$ is the selected subset and X_j is the attribute under study. Then, at learning time $P(X_j|C)$ has been estimated.

Test for Fold	instance id.	X_1	$\ldots$	X_m	$Lp(X_1,\ldots,X_m \mid c)$
1	i_{11}	$Lp(i_{11}^{\downarrow X_1} \mid c_1)$	$\ldots$	$Lp(i_{11}^{\downarrow X_m} \mid c_1)$	$Lp(i_{11}^{\downarrow (X_1,\ldots,X_m)} \mid c_1)$
	$\ldots\ldots\ldots$				
	i_{11}	$Lp(i_{11}^{\downarrow X_1} \mid c_c)$	$\ldots$	$Lp(i_{11}^{\downarrow X_m} \mid c_c)$	$Lp(i_{11}^{\downarrow (X_1,\ldots,X_m)} \mid c_c)$
	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
	i_{1q}	$Lp(i_{1q}^{\downarrow X_1} \mid c_1)$	$\ldots$	$Lp(i_{1q}^{\downarrow X_m} \mid c_1)$	$Lp(i_{1q}^{\downarrow (X_1,\ldots,X_m)} \mid c_1)$
	$\ldots\ldots\ldots$				
	i_{1q}	$Lp(i_{1q}^{\downarrow X_1} \mid c_c)$	$\ldots$	$Lp(i_{1q}^{\downarrow X_m} \mid c_c)$	$Lp(i_{1q}^{\downarrow (X_1,\ldots,X_m)} \mid c_c)$
$\ldots$	$\ldots\ldots\ldots$				
5	i_{51}	$Lp(i_{51}^{\downarrow X_1} \mid c_1)$	$\ldots$	$Lp(i_{51}^{\downarrow X_m} \mid c_1)$	$Lp(i_{51}^{\downarrow (X_1,\ldots,X_m)} \mid c_1)$
	$\ldots\ldots\ldots$				
	i_{51}	$Lp(i_{51}^{\downarrow X_1} \mid c_c)$	$\ldots$	$Lp(i_{51}^{\downarrow X_m} \mid c_c)$	$Lp(i_{51}^{\downarrow (X_1,\ldots,X_m)} \mid c_c)$
	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
	i_{5q}	$Lp(i_{5q}^{\downarrow X_1} \mid c_1)$	$\ldots$	$Lp(i_{5q}^{\downarrow X_m} \mid c_1)$	$Lp(i_{5q}^{\downarrow (X_1,\ldots,X_m)} \mid c_1)$
	$\ldots\ldots\ldots$				
	i_{5q}	$Lp(i_{5q}^{\downarrow X_1} \mid c_c)$	$\ldots$	$Lp(i_{5q}^{\downarrow X_m} \mid c_c)$	$Lp(i_{5q}^{\downarrow (X_1,\ldots,X_m)} \mid c_c)$

Table 3.15: Table stored in memory for incremental classification

Now, we compute a vector (column) as in Table 3.15 for X_j, containing $log(i_{kl}^{\downarrow X_j}$ for each state of C. Then we can compute the column $Lp(i_{kl}^{\downarrow(X_1,\ldots,X_m,X_j)}|C)$ just by adding the content of the last column in the table with the vector recently computed $Lp(X_j|C)$. Thus, for each new variable the complexity of this step is $\mathcal{O}(2ct)$ or just $\mathcal{O}(ct)$ avoiding the constant. Therefore, the global complexity is $\mathcal{O}(ctn)$. With respect to table updating, if the variable under study X_j is not included in S then no modification is carried out over the table, otherwise, the new vector $Lp(X_j|C)$ is added and the last column replaced by $Lp(i_{kl}^{\downarrow(X_1,\ldots,X_m,X_j)}|C)$. Because these two vectors have been previously computed, only pointer operations are required to add and replace the two columns involved.

- IWSS$_r$. Now, besides carrying out the same operations as before, m additional vectors must be computed, each one corresponding to the replacement of a variable $X_r \in S$ by X_j. Computing each one of these vectors is done in the following way:

$$Lp(i_{kl}^{\downarrow(X_1,\ldots,X_r,\ldots,X_m,X_j)}|C) =$$
$$Lp(i_{kl}^{\downarrow(X_1,\ldots,X_r,\ldots,X_m)}|C) - Lp(i_{kl}^{\downarrow X_r}|C) + Lp(i_{kl}^{\downarrow X_j}|C)$$

  Thus, only $Lp(X_j|C)$ and the information (columns) contained in the table are used, and so $m + 2$ columns are computed instead of 2 as in IWSS, and the complexity order is $\mathcal{O}(mct+ct)$, which leads to a global complexity of $\mathcal{O}(nmct+nct)$. Notice that as the search is best choice-driven, there is no need to maintain all the vectors simultaneously in memory.

Finally, it is obvious that space complexity during classification time is higher than in the *black box* approach where only the statistics for the instance being classified are maintained simultaneously in memory. Now, space complexity is $\mathcal{O}(tc(m + 3))$ in IWSS and $\mathcal{O}(tc(m + 5))$ in IWSS$_r$ because $m + 3$ (or $m + 5$) columns must be simultaneously in memory for IWSS (IWSS$_r$). As in most cases $m \geq 3$ and $m \geq 5$, the previous orders were reduced to $\mathcal{O}(2tcm)$ or simply to $\mathcal{O}(tcm)$. Therefore, the memory requirements increase, but as $m << n$, in practice the space requirement is easily affordable in today's computers.

Table 3.16 shows a summary of the complexity orders. The main conclusion is that in the case of IWSS the embedded method is m times faster on average (n in the worst case) than the black box approach, both in learning and classification. With respect to

Table 3.16: Summary of the complexity orders (average and worst case) for blackbox and embedded NB

	IWSS		IWSS$_r$		
	average	worst	average	worst	
Black Box	$\mathcal{O}(tmn)$	$\mathcal{O}(tn^2)$	$\mathcal{O}(tm^2n + tmn)$	$\mathcal{O}(tn^3 + tn^2)$	learning
	$\mathcal{O}(tmnc)$	$\mathcal{O}(tn^2c)$	$\mathcal{O}(tm^2nc + tmnc)$	$\mathcal{O}(tn^3c + tn^2c)$	classification
Embedded	$\mathcal{O}(tn)$	$\mathcal{O}(tn)$	$\mathcal{O}(tn)$	$\mathcal{O}(tn)$	learning
	$\mathcal{O}(tnc)$	$\mathcal{O}(tnc)$	$\mathcal{O}(tncm + tnc)$	$\mathcal{O}(tn^2c + tnc)$	classification

IWSSr, the reduction in classification is the same but a greater saving is obtained in the learning stage, while we get linear complexity for the number of selected variables (m) instead of cubic (m^3). Therefore, the more attributes are selected, the greater the gain in efficiency. On the other hand, the use of early stopping also has its impact, because it reduces from n to approximately $\theta \cdot n$ the number of attributes studied, and this is a variable which appears in all the complexity orders calculated above. The next subsection shows an empirical analysis of the impact of embedding the NB classifier over the set of datasets considered.

3.7.1 Experimental testing of the embedded approach

The previous (big-O) complexity study has shown the expected *theoretical* gain when using the embedded approach instead of the black-box one. However, we know that some implementation details, the use of logs, etc. can reduce or increase gain. In this case, the gain is even more than expected due to the fact that with the embedded approach no dataset management (mainly projections) is needed. Table 3.17 shows the number of times the embedded approach is faster than the black box using Naive Bayes, considering original algorithms and also early stopping with $\theta = 0.2$.

As observed from Table 3.17, the gain is really significant, meaning that a painstaking process requiring hours or days can be solved in seconds or minutes. Table 3.18 shows the ratio between the canonical algorithms (black-box and no early stopping) with respect to the embedded approach plus early stopping. These results give an idea of the improvement obtained when concatenating the two improvements related to efficiency proposed up to here.

Table 3.17: Ratio between the embedded and black-box approaches for Naive Bayes. With and without early stopping ($\theta = 0.2$).

	IWSS^2	**IWSS$_r^2$**	**IWSS$_{r,s}^2$**	**IWSS$_s^2$**
Colon	9.04	46.67	15.92	6.99
Leukemia	56.89	191.08	169.97	22.39
Lymphoma	18.68	123.58	29.87	8.67
DLBCL	31.60	83.63	25.76	16.06
Prostate	85.59	592.65	376.99	41.76
Lung	56.66	240.00	89.98	18.03
GCM	59.83	683.94	550.60	44.68
Arcene	77.43	457.53	234.75	33.62
Madelon	6.03	21.44	7.20	3.78
Dorothea	395.03	2040.40	1661.41	232.20
Dexter	103.43	902.14	360.28	42.46
Gisette	78.04	188.20	154.15	43.66
Mean	**81.52**	**464.27**	**306.41**	**42.86**

Table 3.18: Ratio between the black-box (no early stopping) approach and the embedded one using early stopping ($\theta = 0.2$) for the Naive Bayes classifier.

Dataset	**IWWS**	**IWSS$_r$**
Colon	17.13	56.26
Leukemia	74.59	257.96
Lymphoma	31.46	169.33
DLBCL	37.77	95.96
Prostate	121.61	993.18
Lung	84.92	393.18
GCM	103.39	1427.03
Arcene	112.91	711.23
Madelon	10.08	40.40
Dorothea	968.38	5053.54
Dexter	210.90	2251.70
Gisette	112.63	279.41
Mean	**157.15**	**977.43**

3.8 Discussion of Related Work

Apart from [98] and [33], which introduce the filter (or rank-based) wrapper approach to feature subset selection and thus constitute the basis for the algorithms presented here, there are two recent papers that must be commented.

Nakariyakul and Casasent [80] presented an improved version of floating feature selection that, among other steps, includes what they call *replacing the weak feature*, which is in fact a *type* of replacement. The main difference with the approach in this Chapter lies in the fact that this only tests $m = |S|$ subsets when studying replacement for a given subset S, while in [80] $m \cdot n$ subsets are analyzed because each time a variable X_i of S is tested to be *weak feature*, a new step of forward selection is carried out, taking as starting subset $S - \{X_i\}$, in order to decide which of the remaining variables is the best choice to replace the *weak one* (X_i). Furthermore, as the *replacing of the weak* strategy is inserted in a floating method that combines forward and backward selection phases, the resulting algorithm is quite expensive computationally. As an example, the authors report that for a dataset with 250 attributes and selecting 15 of them, their algorithm (termed IFFS) needs to evaluate 168200 candidate subsets while forward selection only evaluates 7500. Thus, it is clear that although IFSS obtains subsets of very high quality, its use is prohibitive in high-dimensional datasets such as the ones tested in this chapter.

3.9 Conclusions

This Chapter shows four contributions to the improvement of the behavior of the IWSS approach to feature selection. The first provides a study on different relevance criterion for addition of new attributes in the final subset of selected features, concluding than the heuristic condition of $MinFoldersBetter$ is an optimal configuration for IWSS.

The second contribution is related with getting more compact subsets and is implemented by allowing the algorithm to test not only the inclusion of a new attribute (the next one in the ranking) but also the possibility of swapping it with any of the already included attributes. From experiments, IWSS with replacement proves to be a the better choice, because it maintains the accuracy of the SFS and IWSS algorithms, but includes fewer attributes in the selected subset. On the other hand, the number of wrapper evaluations increases with respect to the linearity shown by IWSS, although from the experiments in practice it is far from the theoretical quadratic worst case.

The third contribution in this Chapter directly tackles this problem by stopping the algorithm early, that is, without analyzing all the predictive attributes. From the experiments, is is concluded that the use of early stopping has a significant effect on the efficiency of the algorithms without degrading their performance (both in accuracy and number of selected attributes).

Finally, this Chapter presents an optimization for incremental selection algorithms when using classifiers that allow incremental construction when adding variables, as is the case of Naive Bayes. The idea is to avoid the use of the classifier as a black box, and instead an embedded version of the classifier with the incremental selection algorithm is proposed. The result (both theoretically and experimentally) is a much more efficient incremental selection process.

Chapter 4

An IWSS-based GRASP algorithm for FSS

4.1 Summary

The IWSS approach introduced in Chapter 3 presents one major problem related to the filter ranking: it is deterministic, what derives in two consequences: (a) the search space is highly reduced so it is very easy to fall in a local optimum and (b) features are ordered based on a filter metric which does not capture interaction between features. This Chapter proposes to improve the IWSS search with a GRASP method which (1) makes the ranking stochastic, (2) extends the search space and, not theoretically, might capture (in)dependencies between features.

This proposal to improve the IWSS algorithm is tested using 12 high dimensional databases with extensive experiments and show, not only that it performs better (maintaining accuracy and reducing cardinality of the final subset and number of evaluations) than IWSS but also even better than another feature subset selection algorithms known in the literature.

4.2 Greedy Randomized Adaptive Search Procedure

Following the classification of search algorithms in Chapter 2, the IWSS algorithm falls into the category of a Ranker+Sequential search, where the ranking is created using filter evaluations and the sequential search is incremental (forward) with wrapper evaluations. Thus, IWSS is known as an hybrid FSS algorithms because it mixes filter and wrapper evaluations. Since the created ranking is deterministic, the IWSS approach presents two major problems:

- The incremental search is performed over a fixed ranking, so the search space is highly reduced to a complexity of *O(n)*, and it is very probable to fall in a local optimum.
- Some features ranked near the beginning of the ranking might not be relevant for the class anymore once previous features have been selected and, on the contrary, features ranked at the end of the ranking might become relevant after some features have been selected.

By utilizing GRASP search with the IWSS algorithm in the construction stage, these two problems are to be alleviated by (1) adding randomness at the time of creating the ranking , (2) repeating the process *create ranking + incremental search* a fixed number of times to find several solutions and identify the best one and (3) reducing the size of the ranking instead of using all features.

4.2.1 GRASP

The greedy randomized adaptive search procedure (GRASP) [28] is a meta-heuristic algorithm commonly applied to combinatorial optimization problems. There are two clear stages in a GRASP algorithm:

1. *Construction* phase. In this step a specific (deterministic) heuristic for solving the target problem is taken as a basis for constructing a solution. Thus, starting from the empty set, the algorithm adds elements from all the possible candidates until a solution is obtained. However, in GRASP some randomness is introduced in this step in order to obtain a greedy randomized construction method (also known as semi-greedy heuristic). Thus, instead of choosing the best element at each step of the construction, the algorithm chooses at random from a list of promising ones.
2. *Improving* phase. The solution constructed is taken as the starting point for a local search in order to get an improved solution.

GRASP algorithms run the previous two phases a number of times, working in this way as a multi-start method. Because of the quality provided to the solution by the deterministic greedy heuristic used as a basis, and because of the variability introduced by the randomness added to the process, we can expect to obtain good starting points but with enough variability, and so the multi-start method will eventually find a global optimum.

Recently, [129] proposed to use GRASP for feature selection and compares it with sequential floating algorithms (SFFS, SBFS [89]) and other meta-heuristics like tabu search, genetic algorithms and memetic algorithms. Her conclusion is that tabu search and GRASP outperform the other tested methods. The GRASP algorithm bases the constructive phase on the randomized selection of d attributes according to their *in-group variability* (a filter measure). This process is repeated several times and the subset with the best fitness (computed by using the nearest neighbour approach) is passed as the starting point for the improving step, which is carried out by means of a standard hill-climbing algorithm that uses the same fitness measure and that also looks for subsets having exactly d variables. Then, the previous steps are iterated a number of times. This approach, though returning good results for GRASP with respect to other techniques, is not suitable when dealing with high-dimensional datasets. First, as ot will be seen in the experiments, there is a great variability in the number of variables selected for each dataset, so we cannot fix a number a priori. And second, the number of evaluations (tested candidate subsets) with respect to the number of variables is too high to be used with tens of thousands of variables. In fact, the experiments in [129] only consider datasets having between 18 and 57 variables.

This section describes a proposal for the GRASP algorithm that reduces the number of evaluations to be sub-linear (with respect to $n = |\mathbf{X}|$) and so it is suitable for solving FSS in high-dimensional datasets. The pseudo-code of the proposed algorithm is shown in Figure 4.2, and a detailed description is provided in the next two subsections.

4.2.2 Construction Stage: Randomizing IWSS

The IWSS algorithm is a very efficient algorithm (linear in the number of attributes) that does however have two main disadvantages: (1) it relies on an univariate ranking, so some interesting variables can be judged irrelevant/relevant just because some others have been judged irrelevant/relevant before; and (2) to be sure all the potentially relevant variables have been analyzed, the full ranking must be explored. As an example, let us observe Figure 4.1 which is a plot of the relation between the number of variables in the datasets and the position in the SU-based ranking of the last variable selected by IWSS for the twelve datasets used in our experiments. As we can observe, in 8 out of the 12 datasets, variables after position 100 (the number selected by the linear-forward algorithm [40]) are selected, while the same happens in 6 out of the 12 datasets if we consider the first 10% of the ranking as theshold. The proposal on this chapter ties to (partially) alleviate these two disadvantages by including randomness. The idea is as follows:

- Since the process is to be repeated several times, each iteration only considers a small number of attributes. In this way, the number of wrapper evaluations is drastically reduced.

- The subset of variables considered at each iteration is selected in an *informed* random way. That is, each variable X_i has a selection probability that is proportional to $SU(X_i, C)$. In this way, *good* variables will be selected more often, while *bad* variables have only a small chance. This step is implemented in line 7 of the algorithm in Figure 4.2.

- Once we have selected the variables to be used, they are ranked according to $SU(X_i, C)$ (line 8 of Figure 4.2). Notice, that as only a few variables (with respect to the total) are considered, maintaining a good order is essential, because otherwise given the greedy behaviour of IWSS, very noisy (and large) subsets will be selected.

- Because few variables are considered at each iteration, we give room to the selection of variables that otherwise would always be discarded. For example, let us suppose that our ranking starts with $X_1, X_2, X_3, \ldots$ and that in the score assigned by the wrapper evaluator the following holds: $acc(X_1, X_2) > acc(X_1)$; $acc(X_1, X_2, X_3) < acc(X_1, X_2)$, and $acc(X_1, X_3) > acc(X_1, X_2)$. Then, if we consider this ranking, IWSS will always include the suboptimal selection (X_1, X_2) instead of (X_1, X_3), which is better. However, because of the (pseudo)random selection of small subsets, it could happen than X_2 is not selected in some iterations, and so (X_1, X_3) has its opportunity.

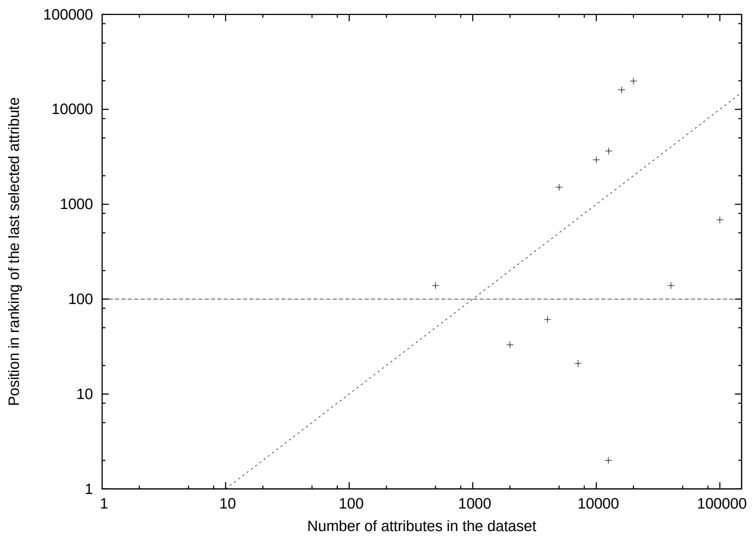

Figure 4.1: Relation between the number of attributes in the dataset and the position of the last attribute selected by IWSS.

4.2.3 Improving step

As mentioned above, the usual improving step in a GRASP algorithm takes the solution found in the constructive step, $\mathcal{S}$, and tries to improve it by using a local search (e.g. Hill-Climbing). However, given our goal of developing a non-expensive-CPU algorithm for high dimensional datasets, proceeding in this standard way is not possible. Notice that a Hill-Climbing algorithm needs exactly n wrapper evaluations just in its first iteration, because it tries to include all the non-used variables one by one, and to remove the already selected variables in the same way. Because we are dealing with thousands of variables, the requirements of such a local optimizer are too high. Therefore, we must think in of a different way of improving.

Our idea is based on the fact that an FSS problem can be viewed as a bi-objective problem, where two parameters are considered: the cardinality of the selected subset and the accuracy provided by such a subset. Taking this into account we can give the following definition:

Definition 1 *Given two candidate subsets $\mathcal{S}_1$ and $\mathcal{S}_2$, we say that $\mathcal{S}_1$ dominates $\mathcal{S}_2$ if $|\mathcal{S}_1| \leq |\mathcal{S}_2|$ and $acc(\mathcal{S}_1) \triangleright^1 acc(\mathcal{S}_2)$. Otherwise we say that $\mathcal{S}_2$ is non-dominated by $\mathcal{S}_1$.*

[1]See Section 3.2 for deep explanation of the $\triangleright$ comparator.

```
In    $\mathbf{T}$: training set; $M$: filter measure; $\mathcal{C}$ classifier algorithm;
      $size$ number of variables to consider at each iteration;
      $numIt$: number of iterations; improving method
Out   $\mathcal{S}$ // The selected subset

      // initialization
1     $NDS \leftarrow \emptyset$
2     for each $X_i \in \mathbf{X}$
3        scores[i]=$M_{\mathbf{T}}(X_i$, C)      // e.g. $SU(X_i, C)$
4     for each $X_i \in \mathbf{X}$      // Prob. of selecting each $X_i$
5        probSel[i]= $scores[i] / \sum_{j=1}^{n} scores[j]$
      // GRASP
6     for it=1 to numIt
         // constructive step
7        $subset \leftarrow$ sample $size$ variables from $\mathbf{X}$ without replacement by using $probSel[\,]$
8        $R[\,] \leftarrow$ create a rank for variables in $subset$ by using $scores[\,]$

9        $\mathcal{S}$ = $\{R[1]\}$      // $\mathcal{S}$ will contain the solution obtained by IWSS
10       BestData = evaluate($\mathcal{C}, \mathcal{S}, \mathbf{T}$)
11       for i = 2 to $R.size()$
12          $\mathcal{S}_{aux}$ = $\mathcal{S} \cup \{R[i]\}$
13          AuxData = evaluate($\mathcal{C}, \mathcal{S}_{aux}, \mathbf{T}$)
14          if (AuxData $\triangleright$ BestData) then
15             $\mathcal{S}$ = $\mathcal{S}_{aux}$
16             BestData = AuxData

         // improving step
17       if (update($NDS$,$\mathcal{S}$)) then
18          $\mathbf{X_{nds}} \leftarrow \cup_{\mathcal{S}_i \in NDS} \mathcal{S}_i$
19          $\mathcal{S}' \leftarrow$ runImprovingMethod($\mathbf{X_{nds}}$, $\mathcal{S}$, $\mathcal{C}$,$\mathbf{T}$)
20          update($NDS$,$\mathcal{S}'$)

21    return all or best solution(s) in $NDS$
```

Figure 4.2: Proposed GRASP algorithm for FSS.

```
In   $NDS$: the set of non dominated solutions.
     $sol$: the candidate solution to be studied.
Out  true if $NDS$ is modified, false otherwise
     parameter $NDS$ is modified
1    If $sol$ is dominated by any $s \in NDS$ then return false
2    else
3      delete from $NDS$ all solutions dominated by $sol$
4      include $sol$ in $NDS$
5      return true
```

Figure 4.3: Auxiliary function update(NDS,sol).

Thus, if we have two different solutions $sol_1 = \langle\{X_1, X_2\}, 0.9, (f_1^1, \ldots, f_5^1)\rangle$ and $sol_2 = \langle\{X_1, X_3, X_4\}, 0.92, (f_1^2, \ldots, f_5^2)\rangle$, where the first component is the subset of selected variables, the second one is the average accuracy over the five folders, and f_i^j is the accuracy in folder i for solution j, which one is better?. Perhaps the correct answer depends on some context, but without extra information, neither sol_1 dominates sol_2 nor does sol_2 dominate sol_1, so it is difficult to decide.

The proposed algorithm will maintain a set of non-dominated solutions (NDS) found during the search. Thus, each time a new solution is provided by the constructive step, the NDS set is updated by using it (function update in Figure 4.3). Then, as we can expect solutions inside NDS to be of good quality, a pool is made with all the variables contained in the non-dominated solutions: $\mathbf{X_{nds}}$. Finally, the improving step consists of running an FSS algorithm whose search space is limited to $\mathcal{P}(\mathbf{X_{nds}})$. In particular, the following ones are used:

- Hill-Climbing. The classical Hill-Climbing algorithm, taking as starting point the solution $\mathcal{S}$ and $\mathbf{X_{nds}}$ as the list of possible attributes. The neighbourhood used is formed by all the subsets having the Hamming distance equals to 1 with respect to the current solution. In this way, the number of evaluations per iteration is $|\mathbf{X_{nds}}|$.

- IWSS. This is the same algorithm (see Chapter 3) used in the constructive phase (lines 8-16 in Figure 4.2), but now it is limited to the variables in $\mathbf{X_{nds}}$. The number of evaluations is exactly $|\mathbf{X_{nds}}|$.

- IWSSr. This algorithm consists of an enhancement of IWSS by adding the operation of replacement (see Chapter 3). Thus, when an attribute ranked in position i is analyzed, not only is its inclusion studied but also its interchange with any of the variables already included in $\mathcal{S}$. That way, the algorithm can retract some of its previous decisions, that is, a previously selected variable can become useless after adding some others. As shown in Chapter 3, this algorithm obtains more compact subsets than IWSS. In the worst case IWSSr will need $\mathcal{O}(|\mathbf{X_{nds}}|)$ evaluations, but in practice the exponent reduces to 1.2-1.3.

- SFS. The classical Sequential Forward Selection [60] (Chapter 2).
- BARS. *Best Agglomerative Ranked Subset* [97] alternates between the construction of a ranking of the available subsets (initially single variables) and a growing heuristic process that obtains all the combinations (by merging) of the first three subsets in the ranking with each one of the remaining ones. After the growing phase, all the subsets with worst accuracy than the current best one are pruned. A new ranking is created and so on. The worst case complexity of BARS is exponential, but in practice it evaluates fewer candidates than IWSSr.

Finally, the solution returned by the *improving* search is used to (again) update NDS (line 20 in Figure 4.2).

Once the GRASP algorithm finishes, we have a set of solutions instead of a single one, so it is possible to choose between returning all of them and letting the user decide which one to use depending on the application context, or we can directly choose one and return it. In this proposal, and in order to compare with standard algorithms that only return one solution, a criterion has been decided that benefits small subsets but without compromising accuracy. Concretely the following procedure is used: (1) rank (from lowest to highest) the solutions in NDS by using the number selected attributes; (2) select the first solution in the ranking as *best*; and (3) run over the ranking and replace *best* by the solution currently analyzed only if it is better according to $\triangleright$ criterion.

4.3 Experiments

This section experimentally tests the GRASP proposal over a set of high-dimensional datasets. Besides analyzing the different improving methods we propose, a comparison with state-of-the-art algorithms is also provided.

4.3.1 Methodology and Test Suite

In pursuit of our goal of dealing with high-dimensional datasets, and in order to obtain reliable conclusions, we selected 12 publicly-obtained, previously presented in Section 3.6.1 .

With respect to FSS subset selection algorithms, taking into account that wrapper evaluation is used and the high cardinality of the datasets, the following four state-of-the-art algorithms are used as a baseline for comparison: IWSS [33; 98], IWSSr (Chapter 3), BARS [97] and Linear Forward Selection (LFS) (Chapter 2).
With respect to our proposal, we test five different instances of the GRASP algorithm depending on the method selected for the improving step: HC, IWSS, IWSSr, SFS and BARS.

In the experiments, all the filter evaluations were performed using *Symmetrical Uncertainty* (SU) and for wrapper evaluation we selected the *Naive Bayes (NB)* algorithm Chapter 2. The reason is two-fold: first, NB is known to be quite sensitive to the presence of redundant and irrelevant predictive attributes, so it is a clear candidate to test FSS; and second, preliminary experiments with other classifiers (decision trees and nearest neighbours) have been carried out and the same conclusions were obtained, so for the sake of clarity and conciseness, only the results for NB are shown here.

4.3.2 Results

The algorithms IWSS and IWSSr do not need parameters. For algorithm LFS we use the code in WEKA [123] and follow the recommendations in [40] (fixed-set with k=100). In the case of BARS we use the values recommended by the authors [97]: k=3 (number of subsets used as seed to form candidates by combination with the remaining ones in the ranking) and ℓ=50% (percentage of the rank to be explored). However, when BARS is used in the improving stage of our GRASP algorithm, ℓ is set to 100% since the search is run over just a few attributes.

Regarding GRASP parameters, we carried out some preliminary tests with different values, and in this study the results with subset size equals to 100 are reported, while two different values (50 and 100) were tested for the number of iterations (multi-starts). The value of subset size can be tuned for each dataset, however, our goal is to carry out an experimentation showing that the results can be generalized to a wide range of datasets without needing a specific tuning. In fact, the tested values are too large for the *smallest* datasets (Madelon and Colon) but we maintain them in the test suite for coherence with previous research.

Experiment 1.-

Table 4.1 shows the results obtained when running the four deterministic algorithms: IWSS, IWSSr, LFS and BARS. In all cases the average over a 10 fold cross validation is reported for *accuracy*, *number of selected variables* and *number of wrapper evaluations*. From the results, and before going into a deeper analysis, it is clear that LFS obtains more compact subsets, needs (by far) fewer evaluations, but also obtains the worst accuracy. Of course, if we increase the percentage of the rank considered, these values would be considerably modified, but as shown in Figure 4.1, the problem is that we do not know in advance the correct percentage of the rank to be used. In the limit it is clear that LFS behaves like SFS, which is comparable in accuracy to IWSSr, but obtains less compact subsets and needs more wrapper evaluations (Chapter 3). With regards to IWSS, it needs fewer evaluations than IWSSr and BARS but includes more features in the selected subset. Regarding IWSSr and BARS, the latter needs fewer evaluations and obtains more compact subsets. However, BARS only explores half the ranking while IWSSr explores all the variables. If all the variables in the rank are used as input in BARS, then the number of evaluations significantly increases (notice that BARS explores $3v$ subsets just in its first iteration, v being the number of considered attributes). Finally, it can also be observed that Gisette dataset is a sort of outlier, because of the number of variables IWSS and IWSSr select in that case.

Experiment 2.-

With regards to the proposed GRASP algorithm, assuming (from preliminary experiments) that 100 is an adequate subset size, let us first investigate its performance by allowing it to do 50 iterations (or multi starts). Table 4.2 shows the obtained results when considering the five improving methods described in Section 4.2.3. Now, the same statistics as before are shown, besides also the number of non-dominated solu-

Dataset	**IWSS**		**IWSSr**		**LFS**		**BARS**	
	Acc	Atts	Acc	Atts	Acc	Atts	Acc	Atts
Colon	80.65	3.8	83.87	2.8	80.65	3.5	85.71	3.0
Leukemia	87.50	2.5	87.50	2.0	93.06	3.3	90.54	2.3
Lymphoma	76.04	8.8	80.21	5.9	71.88	8.9	73.67	6.1
DLBCL	85.11	1.9	80.85	1.8	87.23	3.8	76.00	2.4
Prostate	77.94	11.1	78.68	7.0	69.12	5.3	86.81	3.7
Lung	97.24	2.7	97.24	2.4	96.69	2.6	98.36	3.0
GCM	64.21	36.6	59.47	19.9	56.84	11.2	60.00	15.9
Arcene	70.00	13.4	72.00	6.2	68.00	2.6	74.00	4.9
Madelon	59.85	13.3	60.50	8.0	60.45	5.4	60.30	5.8
Dorothea	93.50	7.4	92.88	6.3	92.38	5.5	93.88	7.3
Dexter	81.00	19.6	83.00	12.9	76.33	8.2	82.67	12.8
Gisette	94.68	112.6	94.07	30.7	89.63	7.9	93.10	13.6
Mean	**80.64**	**19.5**	**80.85**	**8.8**	**78.52**	**5.7**	**81.25**	**6.7**
Number Of Evaluations								
Colon	2000.0		7276.5		450.0		5578.4	
Leukemia	7129.0		21378.4		430.0		14541.0	
Lymphoma	40260.0		27663.0		990.0		15576.0	
DLBCL	4026.0		11134.0		480.0		9476.9	
Prostate	12600.0		94507.8		630.0		22578.8	
Lung	12533.0		42603.5		360.0		24658.1	
GCM	16063.0		309750.4		1220.0		69223.7	
Arcene	10000.0		67359.3		360.0		23785.9	
Madelon	500.0		3818.0		640.0		1403.1	
Dorothea	100000.0		441346.2		650.0		203418.0	
Dexter	20000.0		255027.3		920.0		31153.7	
Gisette	5000.0		137050.7		890.0		9452.2	
Mean	**19175.9**		**118242.9**		**668.3**		**35903.8**	

Table 4.1: Results for the use of four deterministic FSS algorithms.

tions found in each case. Furthermore, because of the stochastic nature of GRASP, the numbers now correspond to the average over 10 independent runs, each one using a 10-fold cross validation.

Experiment 3.-

Here it is investigated whether allowing the algorithm to do a larger number of iterations (multi-starts) introduces a significant improvement. Table 4.2 shows the obtained results.

4.3.3 Statistical Analysis

Statistical tests were performed in order to compare results obtained when running 9 different algorithms: 4 deterministic and the GRASP proposal with the five tested improving methods. In order to be in a position to draw significant conclusions, a multiple-algorithms multiple-datasets comparison is used by performing a Friedman test [37] followed by a post-hoc Holm test [42], as suggested in [23] and using the code provided in [39]. In all the cases the confidence level is set to the standard value $\alpha = 0.05$.

The analysis was performed in three stages. At each stage, a different parameter is analyzed and only the algorithms considered non-different to the control one (marked with the • symbol) passed to the following stage. The three stages are:

1. First, the accuracy of the algorithms is compared. That is, it is not of interest to design an algorithm too fast or that selects compact subsets, but to the price of degrading the classification accuracy.

2. Second, the cardinality of the selected subsets is compared. Once accuracy has been guaranteed, then it seems appropiate to prefer algorithms which select more compact attribute subsets.

3. Finally, the number of wrapper evaluations required by each algorithm is compared.

Tables 4.4 and 4.5 show the results of the analysis for $numIt$ equal to 50 and 100 respectively. The three rows correspond to the three stages previously described.

Dataset	**HC**		**IWSS**		**IWSSr**		**BARS**		**SFS**	
	Acc	Atts	Acc	Atts	Acc	Atts	Acc	Atts	Acc	Atts
Colon	81.13	3.0	79.68	3.4	82.26	3.1	80.00	2.9	80.00	3.5
Leukemia	92.64	2.7	93.75	2.7	91.67	2.8	93.33	2.8	93.61	3.3
Lymphoma	74.90	6.1	77.40	7.5	77.29	6.8	76.35	6.8	78.75	7.5
DLBCL	86.17	2.2	86.60	2.3	87.02	2.1	85.74	2.2	87.66	2.5
Prostate	77.87	5.0	78.68	5.7	77.50	4.6	78.60	5.1	78.16	5.6
Lung	95.69	2.2	95.08	2.2	95.75	2.4	96.02	2.3	96.02	2.4
GCM	55.95	11.7	58.63	19.4	53.63	14.1	57.42	13.3	57.53	20.5
Arcene	80.00	5.7	79.30	6.0	78.50	5.7	79.00	5.2	79.30	6.3
Madelon	60.85	7.6	60.90	7.2	60.85	7.2	60.50	7.6	60.80	7.9
Dorothea	93.36	3.7	93.35	4.2	92.99	3.8	93.50	5.0	93.23	4.4
Dexter	83.47	15.7	83.27	15.5	83.37	15.6	83.07	15.6	83.10	15.8
Gisette	93.06	15.9	94.45	63.3	92.17	31.9	92.41	26.3	93.39	34.1
Mean	**81.26**	**6.8**	**81.76**	**11.6**	**81.08**	**8.3**	**81.33**	**7.9**	**81.80**	**9.5**
Number Of Evaluations										
Colon	5065.2		5018.6		5053.6		5081.3		5013.7	
Leukemia	5472.0		5117.5		5333.3		5761.9		5173.9	
Lymphoma	5608.6		5287.9		6076.6		7343.4		5411.8	
DLBCL	5206.1		5063.2		5152.6		5272.9		5064.8	
Prostate	5252.4		5135.5		5283.2		5361.3		5248.8	
Lung	5940.1		5243.6		5802.5		6781.9		5397.2	
GCM	6150.9		5519.6		7090.1		8669.2		7632.9	
Arcene	5524.3		5203.3		5649.4		5755.4		5392.6	
Madelon	5076.8		5016.9		5059.9		5083.8		5004.4	
Dorothea	5172.2		5075.9		5227.2		5502.8		5147.5	
Dexter	5543.2		5061.8		5421.7		5760.3		5019.5	
Gisette	7206.2		6549.3		10437.5		11511.7		9651.9	
Mean	**5601.5**		**5274.4**		**5965.6**		**6490.5**		**5763.3**	
Number Of Non-Dominated Solutions										
Colon	3.0		2.7		2.7		3.0		3.1	
Leukemia	6.7		6.7		6.4		6.7		6.8	
Lymphoma	4.2		5.7		5.6		5.4		6.4	
DLBCL	5.9		6.0		6.1		6.2		6.1	
Prostate	4.2		5.2		4.0		4.6		5.0	
Lung	9.0		9.4		9.6		9.5		9.7	
GCM	3.4		7.1		5.0		4.1		7.2	
Arcene	5.2		6.1		5.5		5.1		6.9	
Madelon	2.2		2.4		2.5		2.6		1.4	
Dorothea	4.4		4.4		4.2		5.3		5.4	
Dexter	3.9		3.1		3.8		4.6		3.1	
Gisette	4.1		14.3		7.9		7.2		6.9	
Mean	**4.7**		**6.1**		**5.3**		**5.3**		**5.7**	

Table 4.2: Mean Results for the Grasp-based FSS algorithm with $numIt = 50$.

Dataset	**HC**		**IWSS**		**IWSSr**		**BARS**		**SFS**	
	Acc	Atts	Acc	Atts	Acc	Atts	Acc	Atts	Acc	Atts
Colon	80.97	3.1	80.32	3.3	80.97	3.3	80.81	2.9	80.16	3.5
Leukemia	93.33	2.9	94.31	3.0	94.44	3.2	94.58	3.0	94.31	3.4
Lymphoma	76.15	6.1	77.81	7.2	76.25	7.0	78.33	7.0	78.54	7.8
DLBCL	86.81	2.3	86.38	2.2	87.23	2.3	89.36	2.3	87.45	2.4
Prostate	78.24	5.0	79.63	5.6	78.01	4.7	80.22	5.4	78.97	5.8
Lung	95.58	2.5	95.91	2.3	96.13	2.3	95.69	2.4	96.96	2.7
GCM	56.32	11.5	58.95	20.6	54.42	14.8	58.79	13.5	59.68	20.3
Arcene	80.20	5.6	80.40	6.1	80.30	5.7	80.80	5.8	80.20	6.8
Madelon	60.85	7.6	60.90	7.2	60.85	7.2	60.50	7.6	60.80	7.9
Dorothea	93.34	3.8	93.06	4.2	92.95	3.9	93.54	5.5	93.31	4.5
Dexter	83.63	15.7	83.27	15.6	83.30	15.6	83.20	15.6	83.17	15.9
Gisette	93.17	15.8	94.59	68.3	92.54	32.7	92.56	26.5	93.57	29.8
Mean	**81.55**	**6.8**	**82.13**	**12.1**	**81.45**	**8.6**	**82.37**	**8.1**	**82.26**	**9.2**
Number Of Evaluations										
Colon	10103.3		10026.9		10085.7		10116.7		10020.2	
Leukemia	10804.9		10207.8		10625.4		11295.1		10318.5	
Lymphoma	11122.2		10489.7		11912.2		14549.3		10793.3	
DLBCL	10359.8		10099.4		10230.8		10444.3		10110.8	
Prostate	10381.2		10185.6		10349.7		10519.2		10292.2	
Lung	11839.5		10446.1		11493.7		13292.1		10809.8	
GCM	11407.5		10800.7		12996.5		15093.9		14134.0	
Arcene	10838.7		10330.4		10900.9		11339.5		10645.4	
Madelon	10076.8		10016.9		10059.9		10083.8		10004.4	
Dorothea	10283.9		10111.1		10279.0		10912.4		10200.7	
Dexter	10530.4		10068.4		10462.5		10847.8		10023.9	
Gisette	12691.2		12299.3		17846.9		19349.0		16367.6	
Mean	**10869.9**		**10423.5**		**11436.9**		**12320.2**		**11143.4**	
Number Of Non-Dominated Solutions										
Colon	3.4		3.1		3.2		3.3		3.5	
Leukemia	8.4		8.2		8.0		8.4		8.7	
Lymphoma	5.0		6.9		6.8		6.4		8.2	
DLBCL	6.8		7.0		6.9		7.0		7.2	
Prostate	4.8		5.4		4.6		5.2		5.6	
Lung	11.8		11.7		11.6		11.8		11.9	
GCM	3.8		8.5		5.5		4.9		7.5	
Arcene	6.4		7.6		6.6		6.5		8.2	
Madelon	2.2		2.4		2.5		2.6		1.4	
Dorothea	5.4		5.1		5.3		6.6		6.1	
Dexter	3.8		3.3		3.9		4.8		3.1	
Gisette	5.0		16.5		8.9		7.9		7.6	
Mean	**5.6**		**7.1**		**6.2**		**6.3**		**6.6**	

Table 4.3: Mean Results for the Grasp-based FSS algorithm with $numIt = 100$.

Deterministic algorithms are denoted by its name in boldface, while for the GRASP approach (five most right columns) the algorithm is denoted by the name of the used improving method, followed by a $*$ superscript.

Stage	**IWSS**	**IWSSr**	**LFS**	**BARS**	HC^*	$IWSS^*$	$IWSSr^*$	$BARS^*$	SFS^*
Acc	~~80.64~~	80.85	~~78.52~~	81.25	81.26	●81.76	81.08	81.33	81.80
Atts		8.8		6.7	●6.8	~~11.6~~	8.3	7.9	~~9.5~~
Evals		~~118242.9~~		~~35903.8~~	●5601.4		~~5965.6~~	~~6490.5~~	

Table 4.4: Statistical Tests for $numIt = 50$.

Stage	**IWSS**	**IWSSr**	**LFS**	**BARS**	HC^*	$IWSS^*$	$IWSSr^*$	$BARS^*$	SFS^*
Acc	~~80.64~~	80.85	~~78.52~~	81.25	81.55	82.13	81.45	●82.37	82.26
Atts		8.8		●6.7	6.8	~~12.1~~	8.6	8.1	~~9.2~~
Evals		~~118242.9~~		~~35903.8~~	10869.9		●11436.9	~~12320.2~~	

Table 4.5: Statistical Tests for $numIt = 100$.

From the tables, the conclusions are pretty clear, and are basically the same in both cases (50 and 100 iterations):

- None of the instances of the GRASP algorithm can be discarded by accuracy.
- LFS and IWSS are the worst algorithms in accuracy.
- Using SFS and IWSS as improving stage yields larger selected subsets than the remaining algorithms.
- With respect to the number of evaluations, the GRASP approaches clearly improve IWSS$_r$ and BARS.
- Between the three survival GRASP-approaches (HC*, BARS* and IWSSr*) BARS* is always tested as significantly worse when using the number of wrapper evaluations as parameter. This may seem strange according to the behaviour of BARS when used as a deterministic algorithm, however it has a clear explanation. The good performance of BARS with respect to the number of evaluations is due to the pruning of subsets that are worse than the current best, however, now the

pool of available attributes is formed by very good attributes (those in the non-dominated solutions), therefore the subsets created by using these attributes are also very good and only a few of them are pruned, increasing in this way the number of evaluations carried out by BARS in the improving stage.

It is also interesting to remark that the cases in which GRASP(HC*) needs more iterations (GCM and Gisette) are also those cases in which the use of HC as improving method obtains better results with respect to the cardinality of the selected subset. The quality (few number of variables) in these two cases, also has an impact in the existence of a small number of non-dominated solutions.

Finally, the same statistical study was conducted using as parameters accuracy and cardinality of the selected subset, and using as input the eight survival algorithms: IWSS$_r$, BARS, HC*(numIt=50), BARS*(numIt=50), IWSS*(numIt=50), HC*(numIt=100), BARS*(numIt=100) and IWSS*(numIt=100). The result was that no algorithm in this set is found to be significantly worse with respect to those two parameters.

Therefore, a clear recommendation supported by these experiments is to use the GRASP algorithm with 50 iterations or multi-starts, and using HC (or IWSSr) as improving method.

4.3.4 Complexity Order

For the sake of completeness, the number of evaluations of the 9 tested algorithms over the 12 used datasets was used to estimate their in-practice complexity order $\mathcal{O}(n^x)$. The value of x has been computed as the one minimizing the mean square error with respect to the actual number of evaluations for the 12 datasets. Table 4.6 shows the results rounded to two decimal digits.

With respect to the deterministic algorithms, letting out IWSS which is linear and LFS that is sub-linear but whose complexity is controlled by the portion of attributes used, it is interesting to see that IWSSr and BARS, whose worst case complexity is quadratic and exponential respectively, have a better behaviour in practice with $\mathcal{O}(n^{1.13})$ and $\mathcal{O}(n^{1.06})$ respectively. With respect to the GRASP algorithms, the number of evaluations is $numIt \cdot subsetSize + \epsilon$, ϵ being the number of evaluations carried out in the improving stage. Anyway, for comparison reasons their complexity order $\mathcal{O}(n^x)$ is also estimated, and in all the cases the fitted complexity is sub-linear. Concretely x

	IWSS	**IWSSr**	**LFS**	**BARS**	
min	1.00	1.12	0.56	1.04	
fitted	1.00	1.13	0.64	1.06	
max	1.00	1.39	1.04	1.17	
$numIt = 50$	HC^*	$IWSS^*$	$IWSSr^*$	$BARS^*$	SFS^*
min	0.68	0.74	0.74	0.75	0.74
fitted	0.76	0.86	0.86	0.87	0.85
max	1.37	1.37	1.37	1.37	1.37
$numIt = 100$	HC^*	$IWSS^*$	$IWSSr^*$	$BARS^*$	SFS^*
min	0.80	0.80	0.80	0.81	0.80
fitted	0.87	0.87	0.87	0.87	0.87
max	1.48	1.48	1.48	1.48	1.48

Table 4.6: Complexity Order $\mathcal{O}(n^x)$ of the studied algorithms.

is between 0.76 and 0.87 when $numIt = 50$ and is 0.87 for the five GRASP instances when $numIt = 100$ (the difference in this case is in the third decimal digit).

4.4 Conclusions and Future Work

This chapter has presented a GRASP-based algorithm for feature subset selection in high dimensional datasets. The main goal is to maintain the performance (accuracy and degree of reduction in the number of selected attributes) with respect to other state-of-the-art FSS algorithms designed for this problem, but with the advantage of needing a significantly smaller number of wrapper evaluations. To do this, the proposed GRASP algorithm only uses a small fraction of the available attributes in each iteration, which are selected in a pseudo-random way, that is, more promising attributes have more chance to be selected. Another novelty lies in the improving stage, which instead of only improving the last solution, forces cooperation between all the previously found *non-dominated solutions* by making a common pool with the variables they contain and running a FSS algorithm over them. As a result, this chapter's proposal has obtained a highly competitive algorithm for this problem maintaining the performance of state-of-the-art deterministic algorithms but in sub-linear number of wrapper evaluations. From the different GRASP instances tested, it is recommended to use those running

Hill-Climbing or IWSSr in the improving stage.

Chapter 5

Improvement of Incremental Wrapper Selection Algorithms by Re-Ranking

5.1 Summary

As mentioned in Chapter 2, a family of hybrid selection algorithms have recently appeared in the literature: based on a filter ranking, they perform an incremental wrapper selection over that ranking, as shown in Chapter 3.

Though working fine, these methods still have their own problems: (1) they rely on a univariate ranking that does not take into account interaction between the variables already included in the selected subset and the remaining ones; and (2) because of this, a great portion of the rank must be explored, which means that the number of wrapper evaluations can still be too large.

This Chapter presents a proposal for working incrementally at two levels: *block*-level and *attribute*-level. *Block* refers to a set of consecutive attributes in the ranking. Thus, a hybrid algorithm starts with the first block of the ranking and uses it for attribute-level incremental subset selection. Once the block has been analyzed, the remaining attributes are re-ranked by taking into account the current selected subset. Then the process continues with the first block in the new ranking and so on. The method stops when the exploration of a new block does not produce any change in the selected subset. Experiments shown in this chapter for such a proposal uses a filter re-ranking based on conditional mutual information, and the results show that the re-ranking proposal drastically reduces the number of wrapper evaluations without degrading the quality of the subset obtained; in fact, it achieves the same accuracy with a reduction in the number of selected attributes.

5.2 Re-ranking in Hybrid Incremental Wrapper Selection Algorithms

The idea behind re-ranking is to improve the efficiency of the so-called hybrid filter-wrapper FSS algorithms. To do this, the aim is to *drastically* reduce the number of wrapper evaluations by increasing the number of the filter evaluations carried out. This proposal is based on working incrementally, not only at the attribute level, but also at the *block* or *set* of attributes level, taking into account the selected subset $\mathcal{S}$) in the previous blocks. Thus, the selection algorithm starts by using a filter measure to rank the attributes, then an incremental wrapper algorithm $\mathcal{A}$ is applied but only over the first *block*, that is, over the first B ranked attributes. Let $\mathcal{S}$ be the subset of attributes selected from this first block. Then, a new ranking is computed over the remaining attributes but taking into account the already selected ones ($\mathcal{S}$). Then, algorithm $\mathcal{A}$ is run again over the first block in this new ranking but initializing the selected subset to $\mathcal{S}$ instead of $\emptyset$. This process is iterated until no modification in the selected subset is obtained. As shown in experiments in section 5.3.2, the number of *re-ranks* carried out is very small, and so only a small percentage of attributes is explored, which leads to a great reduction in wrapper evaluations (and so in CPU time) but without decreasing the accuracy of the output obtained and there is even a reduction in the size of the selected subset.

Some of the incremental algorithms described in the literature use the ranking just to get the first k variables and then launch a more sophisticated method over a smaller search space. LinearForward [40] and BARS [97] take this decision at the initial stage and in a static way, while IWSS$_r$ and IWSS$_{rs}$ (Chapter 3) take this decision in a dynamic way, adjusting the number of attributes to study each time $\mathcal{S}$ is modified. The main criticism of this behaviour is that the ranking is based on the *individual* merit of each variable with respect to the class, but it does not take into account possible interactions between the attributes. That is, if we have a subset of attributes $X_1, \ldots, X_n$ individually highly correlated with the class but (perhaps) also correlated among each other, then *all* these variables will be in the first positions of the ranking, although only one (or few) of them is likely to be selected at the wrapper stage. In order to alleviate this problem, the usual choice is to select a large number of variables (e.g. 20% - 50%), although this decision implies that a great number of wrapper evaluations have to be carried out. On the other hand, there could be attributes that are marginally uncorrelated with the class, but conditionally correlated with the class given some other attribute(s). In this case, those variables will be ranked in the last positions and so the only way to explore them is to use the full ranking.

Thus the proposed improvement to hybrid incremental algorithms is to use *re-ranking* as a way to overcome the two problems identified above. The idea is to work by using *blocks* (subsets) of variables computed from the ranking, but instead of always using the initial (univariate) ranking we propose to re-rank the remaining attributes by taking into account the current selected subset $\mathcal{S}$. In this way: (1) attributes correlated with $\mathcal{S}$ will be placed at the end of the new ranking because they add nothing to the class once we know the value of variables in $\mathcal{S}$; and (2) variables that are conditionally correlated with the class will be placed early in the ranking if the conditional correlation is due to variables included in $\mathcal{S}$.

The algorithm for *re-ranking-based* incremental selection is shown in Figure 5.1. The following points should be mentioned:

- *Selection algorithm.* As for the selection algorithm, any incremental one can be used. In the experiments section of this Chapter several selection algorithms are tested: IWSS, IWSS$_r$, SFS and BARS. The only modifications needed are: (1) the algorithm is seeded with an initial selected subset; and (2) there is no need to compute the ranking (if it were necessary) because it is received as parameter $\mathcal{B}$.

- *Stop criterion.* As can be observed, the algorithm stops when analyzing a new block does not produce a modification in the selected subset, that is, it returns the same subset received as seed. This is an interesting point because there is no need to decide in advance the number of attributes to explore.

- *Block size.* The block size is a key parameter in this approach. This value must be large enough to give some freedom to the wrapper algorithm, but not so large as to explore a great deal of useless attributes,thus canceling out the advantages of using re-ranking. Several block sizes are tested in the experiments at the end of this Chapter.

```
In    $\mathbf{T}$ training set, $M$ filter measure, $\mathcal{C}$ classifier, $B$ block size
Out   $\mathcal{S}$    // The selected subset
1     list $R$ = $\{\}$    // The ranking, best attributes first
2     for each predictive attribute $A_i$ in $\mathbf{T}$
3        Score=$M_{\mathbf{T}}$($A_i$, class)
4        insert $A_i$ in $R$ according to Score
5     $sol.\mathcal{S} = \emptyset$    // selected variables
6     $sol.eval = null$    // data about the wrapper evaluation of $sol.\mathcal{S}$
7     $\mathcal{B}$ = first block of size $B$ in $R$    // $\mathcal{B}$ is ordered
8     Remove first $B$ variables from $R$
9     $sol$ = IncrementalSelection($\mathbf{T}$,$\mathcal{B}$,$\mathcal{C}$,$\mathcal{S}$)
10    $continue = true$
11    while $continue$ do
12       $R' = \{\}$
13       for each predictive attribute $A_i$ in $R$
14          Score=$M_{\mathbf{T}}$($A_i$, class $|sol.\mathcal{S}$)
15          insert $A_i$ in $R'$ according to Score
16       $R = R'$
17       $\mathcal{B}$ = first block of size $B$ in $R$    // $\mathcal{B}$ is ordered
18       Remove first $B$ variables from $R$
19       $sol'$ = IncrementalSelection($\mathbf{T}$,$\mathcal{B}$,$\mathcal{C}$,$\mathcal{S}$)
20       if($sol.\mathcal{S}$ == $sol'.\mathcal{S}$) //no new feature selected
21          then $continue = false$
22          else $sol = sol'$
23    return ($sol.\mathcal{S}$)
```

Figure 5.1: Re-Ranking Canonical Algorithm.

- *Re-ranking algorithm score.* In order to build the ranking of the remaining attributes, $\{A_1, \ldots, A_r\}$, but considering the current selected subset, it is necessary to score $M(A_i, C|\mathcal{S})$ for each $i = 1, \ldots, r$. As we know, exact computation of this term is not feasible even for moderate sizes of $\mathcal{S}$ because very large (#instances) training sets would be necessary as well as too much time and space. Of course, if the size of $\mathcal{S}$ grows, then this expression is simply computationally intractable.

In the literature we can find different ways to approximate this score. Then, several approaches will be tested in order to identify which method suits best.

1. **Conditional Mutual Information Maximization (CMIM)**. Based on using conditional mutual information [31], CMIM tries to balance the amount of information present for each candidate attribute A_i and class C, and the fact that this information might have been already caught by some feature $A_j \in \mathcal{S}$. Thus, this method selects features maximizing their mutual information with the class but minimizing their pair-to-pair dependency. In this case, given that we have a selected subset $\mathcal{S}$ and a set of attributes to rank $\{A_1, \ldots, A_r\}$, the merit $M(A_i, C|\mathcal{S})$, $i = 1, \ldots, k$ is computed as:

$$M(A_i, C|\mathcal{S}) = \min_{A_j \in \mathcal{S}} I(A_i; C|A_j)$$

2. **Mutual Information-Based Feature Selection (MIFS)**. Similar to the main idea in CMIM, Battiti presented MIFS in [6]. Thus, Battiti suggests approximating the merit $M(A_i, C|\mathcal{S})$, $i = 1, \ldots, k$ by computing it as:

$$M(A_i, C|\mathcal{S}) = I(A_i; C) - \beta \sum_{Aj \in \mathcal{S}} (A_i; A_j)$$

   Where $\beta \in [0, 1]$ and its commonly suggested value is 0.5.

3. **Max-Relevance and Min-Redundancy (MRMR)**. Peng et. al [84] present an approximation similar to MIFS; in this case, the merit $M(A_i, C|\mathcal{S})$, $i = 1, \ldots, k$ is computed as:

$$M(A_i, C|\mathcal{S}) = I(A_i; C) - \frac{1}{|\mathcal{S}|} \sum_{A_j \in \mathcal{S}} (A_i; A_j)$$

Because this thesis deals with n-ary variables and not only binary ones, in the experiments mutual information $I()$ is replaced by Symmetrical Uncertainty $SU()$. As can be seen, the number of calls to the filter measure at re-ranking time is $r \cdot |\mathbb{S}|$. However, these computations are more than compensated for the extreme reduction in the number of calls to the wrapper evaluator.

5.3 Experiments

The test suite of databases used for the experiments is the same as that used for the experiments in Chapter 3.

5.3.1 Algorithms in Experiments

The goal of these experiments is to test the goodness of adding re-ranking in hybrid incremental subset selection algorithms. To do this, the use of re-ranking is combined with IWSS, IWSS$_r$ (setting $mf = 2$ in both cases), BARS (ϵ=100 and k=3) and SFS, for block sizes $B = 5, 10, 20, 30, 40, 50$.

For comparisons, algorithms IWSS, IWSS$_r$, BARS (ϵ=50 and k=3), FSS and LFS algorithms are also run without re-ranking. Finally, 7 different classifiers will be compared in order to obtain much more general conclusions: Naive Bayes, C4.5, ibK (k=1), AODE, Neural Networks (NN), SVM and TAN (see Chapter 2). In all cases the average over a 10 fold cross validation is reported.

In the case of applying re-ranking to the BARS algorithm, original BARS is not compatible with re-ranking in a simple manner. This happens because after finishing a block, there is no clear decision about what to do with features selected so far. Therefore, in the experiments BARS is adapted to the proposed re-ranking algorithm by adding at the beginning of a new block all features selected by BARS up to the last block.

Another possible way to adapt BARS would be to add selected features in previous blocks to the candidate subset at evaluation time, but this made evaluations too complex in terms of subset cardinality, so such an adaptation was not selected.

5.3.2 Results for Naïve Bayes Classifier

Due to the vast amount of results to report and compare, this section deals only with Naïve Bayes classifier through different feature selection algorithms. Table 5.1 shows accuracy, number of attributes selected and number of evaluations (mean after 10cv classification) using Naïve Bayes classifier and well-known feature selection incremental algorithms applied to 12 datasets.

Table 5.1: Results for several FSS methods using NB classifier.

	SFS		**LFS**		**BARS**		**IWSS**		**IWSSr**	
	Acc	Atts	Acc	Atts	Acc	Atts	Acc	Atts	Acc	Atts
Colon	83.87	5.9	80.65	3.5	85.71	3.0	80.65	3.8	83.87	2.8
Leukemia	87.50	3.2	93.06	3.3	90.54	2.3	87.50	2.5	87.50	2.0
Lymphoma	83.33	7.1	71.88	8.9	73.67	6.1	76.04	8.8	80.21	5.9
DLBCL	80.85	3.6	87.23	3.8	76.00	2.4	85.11	1.9	80.85	1.8
Prostate	75.00	5.4	69.12	5.3	86.81	3.7	77.94	11.1	78.68	7.0
Lung	93.92	2.5	96.69	2.6	98.36	3.0	97.24	2.7	97.24	2.4
GCM	58.42	18.3	56.84	11.2	60.00	15.9	64.21	36.6	59.47	19.9
Arcene	68.00	4.6	68.00	2.6	74.00	4.9	70.00	13.4	72.00	6.2
Madelon	60.75	6.5	60.45	5.4	60.30	5.8	59.85	13.3	60.50	8.0
Dorothea	91.25	13.2	92.38	5.5	93.88	7.3	93.50	7.4	92.88	6.3
Dexter	76.00	13.8	76.33	8.2	82.67	12.8	81.00	19.6	83.00	12.9
Gisette	94.05	26.9	89.63	7.9	93.10	13.6	94.68	112.6	94.07	30.7
Mean	**79.41**	**9.3**	**78.52**	**5.7**	**81.25**	**6.7**	**80.64**	**19.5**	**80.85**	**8.8**
	Evaluations									
Colon	13800.0		450		5578.4		2000.0		7276.5	
Leukemia	29941.8		430		14541.0		7129.0		21378.4	
Lymphoma	32610.6		990		15576.0		40260.0		27663.0	
DLBCL	18519.6		480		9476.90		4026.0		11134.0	
Prostate	80640.0		630		22578.80		12600.0		94507.8	
Lung	43865.5		360		24658.10		12533.0		42603.5	
GCM	310015.9		1220		69223.7		16063.0		309750.4	
Arcene	56000.0		360		23785.9		10000.0		67359.3	
Madelon	3750.0		640		1403.10		500.0		3818.0	
Dorothea	1420000.0		650		203418.0		100000.0		441346.2	
Dexter	296000.0		920		31153.70		20000.0		255027.3	
Gisette	184500.0		890		9452.20		5000.0		137050.7	
Mean	**207470.3**		**668.3**		**35903.82**		**19175.9**		**118242.9**	

Section 5.2 introduced 3 different scores to apply the re-ranking methodology proposed in this Chapter: CMIM, MIFS and MRMR. For the sake of order and clarity, in this section only CMIM will be used; later, experiments using MIFS and MRMR will also be run.

Tables 5.2, 5.3, 5.4 and 5.5 show the results of applying the re-ranking (CMIM-based) methodology to algorithms SFS, BARS, IWSS and IWSSr respectively, for block sizes $B = 5, 10, 20, 30, 40, 50$.

Table 5.2: Results using Naive Bayes classifier, SFS selection algorithm and CMIM-based re-ranking with block sizes B.

DataSet	**SFS**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	83.87	5.9	82.26	2.2	85.48	2.2	83.87	2.4	83.87	2.4	83.87	2.3	83.87	2.3
Leukemia	87.50	3.2	87.50	1.9	87.50	1.9	87.50	1.9	87.50	1.9	87.50	1.9	87.50	2.0
Lymphoma	83.33	7.1	72.92	4.7	75.00	5.6	80.21	5.6	81.25	5.7	78.13	5.9	76.04	5.9
DLBCL	80.85	3.6	87.23	1.5	82.98	1.6	80.85	1.7	80.85	1.7	80.85	1.7	80.85	1.7
Prostate	75.00	5.4	73.53	3.2	75.74	3.4	77.21	4.2	80.88	4.7	80.15	4.7	83.09	4.8
Lung	93.92	2.5	96.69	2.2	96.69	2.2	97.24	2.4	97.24	2.4	97.24	2.4	97.24	2.4
GCM	58.42	18.3	51.58	7.4	53.68	10.3	57.89	10.9	57.89	11.8	60.00	13.4	62.11	14.2
Arcene	68.00	4.6	71.00	2.6	70.00	3.7	72.00	3.8	73.00	4.3	71.00	4.3	69.00	4.3
Madelon	60.75	6.5	61.65	2.0	60.75	3.4	61.25	4.8	60.25	5.9	60.10	5.6	60.50	6.2
Dorothea	91.25	13.2	94.25	3.0	93.25	4.0	93.38	5.0	93.00	5.3	92.88	5.3	92.88	5.3
Dexter	76.00	13.8	82.67	8.5	81.00	10.1	83.00	9.8	82.67	9.7	82.67	9.8	82.67	9.4
Gisette	94.05	26.9	86.20	2.7	88.47	6.4	90.77	10.8	91.62	15.9	92.25	16.5	92.57	17.1
Geom. Mean	*78.51*	*7.1*	*77.82*	*3.0*	*78.16*	*3.8*	*79.54*	*4.4*	*79.93*	*4.7*	*79.68*	*4.8*	*79.85*	*4.9*
Arith. Mean	*79.41*	*9.3*	*78.96*	*3.5*	*79.21*	*4.6*	*80.43*	*5.3*	*80.84*	*6.0*	*80.55*	*6.2*	*80.69*	*6.3*
Test			=	⇓	=	⇓	=	⇑	=	⇑	=	⇑	=	⇑

Table 5.3: Results using Naive Bayes classifier, BARS selection algorithm and CMIM-based re-ranking with block sizes B.

DataSet	**BARS**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
Colon	85.71	3.0	79.03	3.5	83.87	3.4	79.03	3.3	77.42	3.1	79.03	2.9	79.03	3.0
Leukemia	90.54	2.3	91.67	3.1	93.06	3.3	93.06	3.1	93.06	3.1	94.44	3.2	94.44	3.2
Lymphoma	73.67	6.1	71.88	8.3	73.96	7.8	79.17	9.2	78.13	9.1	77.08	9.1	79.17	9.5
DLBCL	76.00	2.4	85.11	3.1	85.11	3.3	74.47	3.4	74.47	3.6	80.85	3.2	76.60	3.3
Prostate	86.81	3.7	76.47	4.0	67.65	5.0	68.38	5.5	70.59	5.2	77.21	5.9	73.53	6.6
Lung	98.36	3.0	95.58	3.1	97.24	3.6	96.13	3.7	96.13	3.6	96.13	3.5	96.69	3.5
GCM	60.00	15.9	48.42	8.9	55.26	11.4	55.26	14.0	59.47	15.7	60.53	17.2	61.05	17.6
Arcene	74.00	4.9	76.00	4.6	81.00	4.6	82.00	6.0	78.00	6.5	83.00	7.3	85.00	6.7
Madelon	60.30	5.8	61.00	2.7	61.30	4.2	61.05	6.3	61.40	7.3	61.20	8.9	61.05	8.7
Dorothea	93.88	7.3	93.88	6.3	93.63	10.3	94.75	9.1	94.25	11.1	94.00	13.4	94.38	17.1
Dexter	82.67	12.8	77.00	6.1	84.67	12.8	82.67	15.5	83.33	15.5	83.67	15.8	84.67	15.2
Gisette	93.10	13.6	87.05	3.8	87.50	5.8	88.98	7.5	92.02	13.6	92.28	14.3	89.72	14.3
Geom. Mean	*80.29*	*5.4*	*77.33*	*4.4*	*79.26*	*5.6*	*78.52*	*6.3*	*78.94*	*6.8*	*80.77*	*7.2*	*80.41*	*7.39*
Arith. Mean	*81.25*	*6.7*	*78.59*	*4.8*	*80.35*	*6.3*	*79.58*	*7.2*	*79.86*	*8.1*	*81.62*	*8.7*	*81.28*	*9.06*
Test			=	=	=	=	=	=	=	⇑	=	⇑	⇑	=

Table 5.4: Results using Naive Bayes classifier, IWSS^2 selection algorithm and CMIM-based re-ranking with block sizes B.

DataSet	**IWSS^2**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	80.6	3.8	80.6	2.8	83.9	3.0	82.3	3.2	82.3	3.3	82.3	3.3	82.3	3.3
Leukemia	87.5	2.5	87.5	2.0	87.5	2.4	87.5	2.4	87.5	2.4	87.5	2.4	87.5	2.5
Lymphoma	76.0	8.8	66.7	6.3	75.0	7.6	76.0	7.9	77.1	8.0	75.0	8.1	77.1	8.2
DLBCL	85.1	1.9	89.4	1.5	87.2	1.6	85.1	1.7	85.1	1.7	85.1	1.7	85.1	1.7
Prostate	77.9	11.1	72.1	4.1	74.3	4.0	77.9	5.6	74.3	7.3	72.1	7.8	74.3	8.0
Lung	97.2	2.7	96.7	2.2	96.7	2.4	97.2	2.7	97.2	2.7	97.2	2.7	97.2	2.7
GCM	64.2	36.6	54.2	12.3	60.0	19.8	62.1	21.4	65.3	22.5	64.2	24.4	64.7	27.4
Arcene	70.0	13.4	70.0	3.5	68.0	5.1	70.0	6.8	70.0	7.0	70.0	7.8	69.0	7.8
Madelon	59.9	13.3	61.3	2.7	60.9	4.8	60.3	7.1	59.8	8.0	60.0	10.1	59.6	11.4
Dorothea	93.5	7.4	93.9	2.8	94.1	3.6	94.4	3.8	94.0	4.3	93.9	4.3	93.8	4.5
Dexter	81.0	19.6	81.7	11.9	83.7	13.1	83.7	14.8	81.3	15.7	83.0	15.2	80.7	14.9
Gisette	94.7	112.6	88.7	18.3	92.3	41.3	93.7	62.6	93.9	69.5	94.4	82.0	94.1	77.2
Geom. Mean	*79.81*	*9.45*	*77.41*	*4.24*	*79.34*	*5.52*	*79.97*	*6.50*	*79.81*	*6.94*	*79.51*	*7.32*	*79.59*	*7.49*
Arith. Mean	*80.6*	*19.5*	*78.5*	*5.9*	*80.3*	*9.1*	*80.9*	*11.7*	*80.6*	*12.7*	*80.4*	*14.2*	*80.4*	*14.1*
Test			=	$\Downarrow$	=	$\Downarrow$	=	$\Downarrow$	=	$\Downarrow$	=	$\Downarrow$	=	$\Downarrow$

Table 5.5: Results using Naive Bayes classifier, IWSS$_r^2$ selection algorithm and CMIM-based re-ranking with block sizes B.

DataSet	**IWSS$_r^2$**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	83.9	2.8	82.3	2.2	85.5	2.2	83.9	2.4	83.9	2.4	83.9	2.3	83.9	2.3
Leukemia	87.5	2.0	87.5	1.9	87.5	1.9	87.5	1.9	87.5	1.9	87.5	1.9	87.5	2.0
Lymphoma	80.2	5.9	72.9	4.7	75.0	5.6	80.2	5.6	81.3	5.7	78.1	5.9	76.0	5.9
DLBCL	80.9	1.8	87.2	1.5	83.0	1.6	80.9	1.7	80.9	1.7	80.9	1.7	80.9	1.7
Prostate	78.7	7.0	73.5	3.2	75.7	3.4	77.2	4.2	80.9	4.7	80.1	4.7	83.1	4.8
Lung	97.2	2.4	96.7	2.2	96.7	2.2	97.2	2.4	97.2	2.4	97.2	2.4	97.2	2.4
GCM	59.5	19.9	51.6	7.4	53.7	10.3	57.9	10.9	57.9	11.8	60.0	13.4	62.1	14.2
Arcene	72.0	6.2	71.0	2.6	70.0	3.7	72.0	3.8	73.0	4.3	71.0	4.3	69.0	4.3
Madelon	60.5	8.0	61.7	2.0	60.8	3.4	61.3	4.8	60.3	5.9	60.1	5.6	60.5	6.2
Dorothea	92.9	6.3	94.3	3.0	93.3	4.0	93.4	5.0	93.0	5.3	92.9	5.3	92.9	5.3
Dexter	83.0	12.9	82.7	8.5	81.0	10.1	83.0	9.8	82.7	9.7	82.7	9.8	82.7	9.4
Gisette	94.1	30.7	86.2	2.7	88.5	6.4	90.8	10.8	91.6	15.9	92.3	16.5	92.6	17.1
Geom. Mean	*79.97*	*6.06*	*77.82*	*3.01*	*78.16*	*3.81*	*79.54*	*4.35*	*79.93*	*4.72*	*79.68*	*4.76*	*79.85*	*4.85*
Arith.Mean	*80.9*	8.8	*79.0*	*3.5*	*79.2*	*4.6*	*80.4*	*5.3*	*80.8*	*6.0*	*80.6*	*6.2*	*80.7*	*6.3*
Test			=	$\Downarrow$	=	$\Downarrow$	=	$\Downarrow$	=	$\Downarrow$	=	$\Downarrow$	=	$\Downarrow$

The last two rows of the tables show the average value (arithmetic and geometric mean) over the 12 datasets and the result of carrying out the non-parametrical Wilcoxon Matched-Pairs Signed-Ranks Test [121] [23] between the original FSS algorithm (second column) and the same algorithm with re-ranking for each block size. Statistical tests are performed with a confidence level of $\alpha = 0.05$, and the result of a test is indicated in the corresponding cell as: $=$ if there is no statistical difference, $\Uparrow$ if the re-rank algorithm returns a significantly larger value, and $\Downarrow$ if the re-rank algorithm returns a significantly smaller value.

As it can be observed, taking as input the averaged results over the twelve datasets, algorithms SFS, IWSS and IWSSr show that there is no difference in accuracy between using re-ranking or not, while a significant reduction is obtained with respect to the size of the selected subset for all block sizes in algorithms IWSS and IWSSr, and up to $B = 10$ for SFS. The adapted BARS with re-ranking performs statistically the same (in terms of both accuracy and number of selected attributes) as the original BARS for low block sizes.

Once it has been stated that the use of re-ranking does not degrade the quality of the obtained output (on the contrary, it gets more compact subsets), let us study the behaviour of the re-ranking algorithm in terms of wrapper evaluations (by far, the most expensive ones). Table 5.6 shows the reduced % of evaluations and re-ranks carried out by SFS, BARS, IWSS and IWSS$_r$ with Naïve Bayes as classifier. The results shown are for the 6 block sizes considered and averaged (arithmetic and geometric mean) over the 12 datasets. As can be observed the reduction with respect to the number of evaluations carried out is really impressive and the number of re-ranks very small. Resulting the the proposed re-ranking algorithm proves to be a very optima choice to improve subsets compactness and evaluations complexity for at least the 4 FSS algorithms tested.

A global comparison can be viewed in Figure 5.2 (note the log-scale on the Y-axis).

5.3.3 Re-Ranking Criteria

Once it has been shown that applying the CMIM-based re-ranking proposal to several feature subset selection algorithms drastically reduces the number of evaluations, maintaining the performance in terms of accuracy, and even reducing the cardinality of the final selected subset for low block sizes B, it is now time to compare CMIM with other criteria in order to find out if there is any difference among them. Thus, 3

Table 5.6: Mean number of evaluations and re-ranks performed over the 12 datasets.

	#evals (arith.)	**#evals (geom.)**	**%arith.**	**%geom.**	**#re-ranks**
B=5	45.43	55.93	99.98	99.91	1.53
B=10	120.03	156.24	99.94	99.75	1.84
B=20	264.76	338.38	99.87	99.47	1.84
B=30	442.74	621.43	99.79	99.02	1.97
B=40	565.14	821.40	99.73	98.71	1.78
B=50	699.00	959.81	99.66	98.49	1.63
SFS (arithm. 207470.28 evals., geom. 63691.78 evals.)					
B=5	146.42	116.82	99.59	99.34	1.53
B=10	502.33	325.07	98.60	98.16	1.84
B=20	1012.50	626.37	97.18	96.45	1.84
B=30	1219.92	940.78	96.60	94.67	1.97
B=40	1614.92	1268.85	95.50	92.82	1.78
B=50	1801.17	1468.86	94.98	91.68	1.63
BARS (arithm. 35903.82 evals., geom. 17663.70 evals.)					
B=5	18.33	15.97	99.90	99.82	2.47
B=10	40.83	33.60	99.79	99.63	2.98
B=20	81.67	66.62	99.57	99.26	3.03
B=30	113.75	95.10	99.41	98.95	2.76
B=40	152.00	126.61	99.21	98.60	2.78
B=50	185.17	157.43	99.03	98.26	2.68
IWSS^2 (arithm. 19175.92 evals., geom. 9027.88)					
B=5	55.93	45.43	99.95	99.91	1.53
B=10	156.24	120.03	99.87	99.76	1.84
B=20	338.38	264.76	99.71	99.47	1.84
B=30	621.43	442.74	99.47	99.11	1.97
B=40	821.40	565.14	99.31	98.87	1.78
B=50	959.81	699.00	99.19	98.60	1.63
IWSS$_r^2$ (arithm. 118242.93 evals., geom. 49877.48)					

re-ranking methods will be compared:

1. CMIM.
2. MIFS.
3. MRMR.

Experiments are run again using the NB classifier and over 12 datasets, reporting results of a 10 cross-validation in Table 5.7, with block sizes $B = 5$ and $B = 10$.

Finally, Table 5.8 shows the results of comparing all criteria re-ranking (besides the original algorithm) for IWSS^2, IWSS$_r^2$, SFS and BARS by performing a Friedman test [37] followed by a post-hoc Holm test [42], as suggested in [23] and using the code

Table 5.7: Results using Naive Bayes classifier, for re-ranking based on CMIM, MIFS and MRMR criteria. Block sizes $B = 5$ and $B = 10$

DataSet	**SFS**		**CMIM$_5$**		**CMIM$_{10}$**		**MIFS$_5$**		**MIFS$_{10}$**		**MRMR$_5$**		**MRMR$_{10}$**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	83.87	5.9	85.48	2.4	82.26	3.9	83.87	2.7	82.26	4.2	85.48	2.1	85.48	2.8
Leukemia	87.50	3.2	91.67	3.3	94.44	3.4	91.67	3.6	94.44	3.2	88.89	2.9	95.83	3.4
Lymphoma	83.33	7.1	80.21	11.0	83.33	13.8	76.04	9.0	76.04	11.5	73.96	8.0	80.21	10.8
DLBCL	80.85	3.6	89.36	3.5	89.36	3.9	91.49	4.4	91.49	4.3	91.49	3.0	91.49	3.2
Prostate	75.00	5.4	77.21	4.1	80.15	4.6	75.00	4.3	76.47	5.7	76.47	3.8	77.21	4.5
Lung	93.92	2.5	95.03	3.6	96.13	3.1	96.13	3.4	96.13	3.0	95.58	2.8	96.13	2.8
GCM	58.42	18.3	55.26	17.9	56.32	18.5	65.26	19.9	70.00	24.5	50.53	13.2	51.58	10.5
Arcene	68.00	4.6	76.00	5.9	76.00	5.9	74.00	6.8	72.00	8.4	78.00	4.8	73.00	7.0
Madelon	60.75	6.5	61.25	3.5	60.80	4.2	61.25	2.4	60.75	3.7	61.25	2.4	60.75	3.7
Dorothea	91.25	13.2	92.75	6.6	92.63	4.5	93.25	6.2	92.63	4.5	93.25	4.5	92.75	4.2
Dexter	76.00	13.8	87.00	15.8	87.67	18.5	80.00	12.0	83.00	13.5	75.00	6.4	80.67	7.7
Gisette	94.05	26.9	88.90	10.5	89.70	18.0	90.83	16.7	89.70	15.8	88.00	9.0	88.10	11.2
Mean	**79.41**	**9.3**	**81.68**	**7.3**	**82.40**	**8.5**	**81.57**	**7.6**	**82.08**	**8.5**	**79.82**	**5.2**	**81.10**	**6.0**
SFS														

DataSet	**IWSS^2**		**CMIM$_5$**		**CMIM$_{10}$**		**MIFS$_5$**		**MIFS$_{10}$**		**MRMR$_5$**		**MRMR$_{10}$**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	80.65	3.8	80.65	2.8	83.87	3.0	80.65	2.8	83.87	2.9	80.65	2.7	83.87	2.9
Leukemia	87.50	2.5	87.50	2.0	87.50	2.4	87.50	2.1	87.50	2.4	87.50	1.8	87.50	2.1
Lymphoma	76.04	8.8	66.67	6.3	75.00	7.6	71.88	7.0	72.92	7.3	64.58	5.1	72.92	5.6
DLBCL	85.11	1.9	89.36	1.5	87.23	1.6	89.36	1.5	87.23	1.6	89.36	1.5	87.23	1.6
Prostate	77.94	11.1	72.06	4.1	74.26	4.0	73.53	4.5	66.91	4.3	70.59	3.2	70.59	3.9
Lung	97.24	2.7	96.69	2.2	96.69	2.4	96.69	2.3	96.69	2.4	96.13	2.1	96.13	2.2
GCM	64.21	36.6	54.21	12.3	60.00	19.8	65.26	14.9	60.53	18.9	49.47	8.9	48.95	8.7
Arcene	70.00	13.4	70.00	3.5	68.00	5.1	73.00	4.4	73.00	7.4	74.00	3.1	73.00	4.9
Madelon	59.85	13.3	61.25	2.7	60.90	4.8	61.15	3.0	60.65	4.7	61.15	3.0	60.65	4.7
Dorothea	93.50	7.4	93.88	2.8	94.13	3.6	93.50	3.4	93.75	3.7	93.50	2.4	93.88	2.9
Dexter	81.00	19.6	81.67	11.9	83.67	13.1	84.00	10.7	83.33	12.5	72.00	5.1	77.67	5.3
Gisette	94.68	112.6	88.67	18.3	92.25	41.3	90.50	14.8	90.23	15.8	88.48	6.0	88.00	8.2
Mean	**80.64**	**19.5**	**78.55**	**5.9**	**80.29**	**9.1**	**80.58**	**6.0**	**79.72**	**7.0**	**77.28**	**3.7**	**78.37**	**4.4**
IWSS^2														

DataSet	**IWSS^2_r**		**CMIM$_5$**		**CMIM$_{10}$**		**MIFS$_5$**		**MIFS$_{10}$**		**MRMR$_5$**		**MRMR$_{10}$**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	83.87	2.8	82.26	2.2	85.48	2.2	82.26	2.0	85.48	2.3	80.65	2.0	80.65	2.3
Leukemia	87.50	2.0	87.50	1.9	87.50	1.9	87.50	2.0	87.50	1.9	87.50	1.8	87.50	1.9
Lymphoma	80.21	5.9	72.92	4.7	75.00	5.6	72.92	5.7	78.13	5.1	67.71	4.1	72.92	4.8
DLBCL	80.85	1.8	87.23	1.5	82.98	1.6	87.23	1.5	82.98	1.6	87.23	1.5	82.98	1.6
Prostate	78.68	7.0	73.53	3.2	75.74	3.4	72.79	3.9	73.53	3.5	72.79	2.9	74.26	3.1
Lung	97.24	2.4	96.69	2.2	96.69	2.2	96.69	2.2	96.69	2.3	96.13	2.1	96.13	2.1
GCM	59.47	19.9	51.58	7.4	53.68	10.3	55.79	9.1	61.58	12.5	48.95	6.5	47.37	8.3
Arcene	72.00	6.2	71.00	2.6	70.00	3.7	70.00	3.3	71.00	3.9	72.00	2.5	70.00	4.0
Madelon	60.50	8.0	61.65	2.0	60.75	3.4	61.10	2.1	60.55	3.6	61.10	2.1	60.55	3.6
Dorothea	92.88	6.3	94.25	3.0	93.25	4.0	93.50	3.7	93.63	5.0	93.75	3.8	92.63	3.9
Dexter	83.00	12.9	82.67	8.5	81.00	10.1	84.00	9.0	82.67	10.0	73.00	5.0	79.33	4.9
Gisette	94.07	30.7	86.20	2.7	88.47	6.4	90.10	10.8	90.02	9.7	88.28	5.8	88.38	7.1
Mean	**80.85**	**8.8**	**78.96**	**3.5**	**79.21**	**4.6**	**79.49**	**4.6**	**80.31**	**5.1**	**77.42**	**3.3**	**77.72**	**4.0**
IWSS^2_R														

DataSet	**BARS**		**CMIM$_5$**		**CMIM$_{10}$**		**MIFS$_5$**		**MIFS$_{10}$**		**MRMR$_5$**		**MRMR$_{10}$**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	85.71	3.0	79.03	3.5	83.87	3.4	82.26	3.1	82.26	3.5	82.26	2.9	82.26	3.1
Leukemia	90.54	2.3	91.67	3.1	93.06	3.3	91.67	3.0	91.67	3.2	91.67	2.2	91.67	3.0
Lymphoma	73.67	6.1	71.88	8.3	73.96	7.8	73.96	6.8	75.00	7.9	69.79	5.5	71.88	5.6
DLBCL	76.00	2.4	85.11	3.1	85.11	3.3	82.98	2.8	93.62	3.4	85.11	2.4	85.11	2.8
Prostate	86.81	3.7	76.47	4.0	67.65	5.0	80.15	3.8	72.06	4.5	75.00	3.6	75.00	3.9
Lung	98.36	3.0	95.58	3.1	97.24	3.6	96.13	3.3	97.24	3.3	96.69	2.6	97.24	3.3
GCM	60.00	15.9	48.42	8.9	55.26	11.4	55.79	10.8	66.32	12.9	48.42	6.9	46.84	7.2
Arcene	74.00	4.9	76.00	4.6	81.00	4.6	82.00	4.5	80.00	5.2	77.00	4.8	79.00	4.2
Madelon	60.30	5.8	61.00	2.7	61.30	4.2	60.70	2.0	60.60	3.8	60.70	2.0	60.60	3.8
Dorothea	93.88	7.3	93.88	6.3	93.63	10.3	93.75	6.2	93.50	7.4	93.38	5.1	93.75	5.3
Dexter	82.67	12.8	77.00	6.1	84.67	12.8	77.67	7.3	79.33	11.1	75.33	6.2	79.33	8.7
Gisette	93.10	13.6	87.05	3.8	87.50	5.8	88.42	5.8	88.69	8.6	88.55	6.3	87.77	8.1
Mean	**81.25**	**6.7**	**78.59**	**4.8**	**80.35**	**6.3**	**80.46**	**5.0**	**81.69**	**6.2**	**78.66**	**4.2**	**79.20**	**4.9**
BARS														

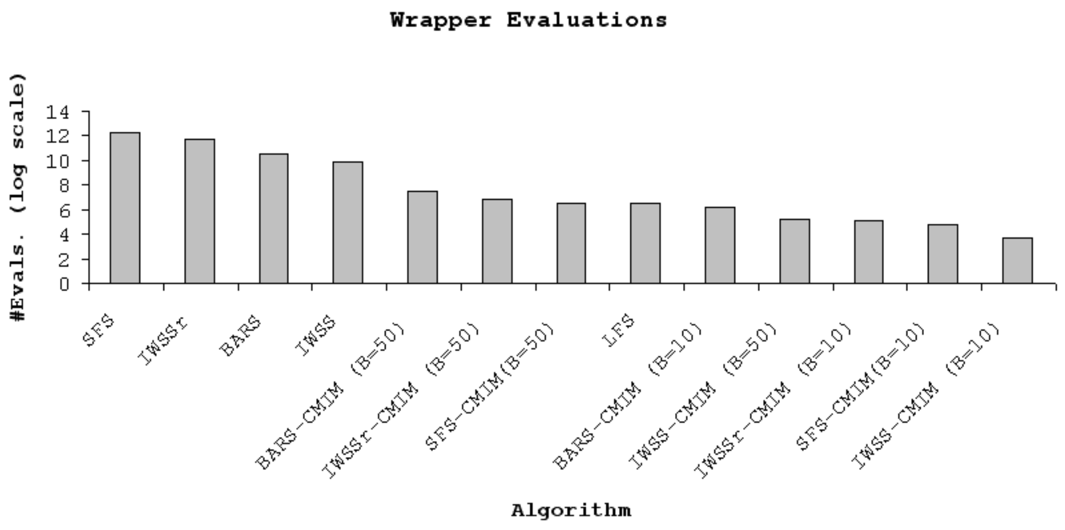

Figure 5.2: Number of Wrapper Evaluations for each selection algorithm and re-ranking.

provided in [39]; the confidence level is set as $\alpha = 0.05$.

Friedman and Holm's tests are run in two stages. At each stage, the control algorithm is marked with the • symbol, and algorithms found to be statistically worse than the control algorithm are crossed out. The two stages are:

1. Tests are run comparing accuracy for all algorithms.

2. Tests are run comparing number of attributes selected for remaining algorithms.

The statistical tests shown in Table 5.8 always keep criterion CMIM with $B = 5$ as one of the best re-ranking criteria. Thus, this is one of the criteria used in the next section in order to compare the effect of re-ranking for several classifiers.

In case the reader is interested in the results for all block sizes and the three re-ranking criteria, these are tabbed in Appendix A.

Table 5.8: Statistical Tests for different Re-ranking Criteria.

Stage	**SFS**	**CMIM**$_5$	**CMIM**$_{10}$	**MIFS**$_5$	**MIFS**$_{10}$	**MRMR**$_5$	MRMR$_{10}$
Acc	~~79.41~~	81.68	●82.40	81.57	82.08	79.82	81.10
Atts		7.3	~~8.5~~	~~7.6~~	~~8.5~~	●5.2	6.0
				SFS			
Stage	**IWSS**^{2}	**CMIM**$_5$	**CMIM**$_{10}$	**MIFS**$_5$	**MIFS**$_{10}$	**MRMR**$_5$	MRMR$_{10}$
Acc	80.64	78.55	●80.29	80.58	79.72	~~77.28~~	78.37
Atts	~~19.5~~	●5.9	~~9.1~~	~~6.0~~	~~7.0~~		4.4
				IWSS^2			
Stage	**IWSS**$_r^2$	**CMIM**$_5$	**CMIM**$_{10}$	**MIFS**$_5$	**MIFS**$_{10}$	**MRMR**$_5$	MRMR$_{10}$
Acc	●80.85	78.96	79.21	79.49	80.31	~~77.42~~	~~77.72~~
Atts	~~8.8~~	●3.5	~~4.6~~	~~4.6~~	~~5.1~~		
				IWSS$_r^2$			
Stage	**BARS**	**CMIM**$_5$	**CMIM**$_{10}$	**MIFS**$_5$	**MIFS**$_{10}$	**MRMR**$_5$	MRMR$_{10}$
Acc	81.25	78.59	~~80.35~~	80.46	81.69	~~78.66~~	79.20
Atts	~~6.7~~	4.8	~~6.3~~	●5.0	~~6.2~~		4.9
				BARS			

5.3.4 Re-Ranking Comparison Among Classifiers

In this section, the effect of re-ranking in $IWSS^2$ and $IWSS^2_r$ is tested across different classifiers, using CMIM as criterion. Since the search algorithms used in comparisons are wrapper-driven, and in the following experiments the classifier is changed, it is interesting to test again all block sizes from 5 to 50.

Results for each database are not tabbed anymore, but the mean of the accuracy and of the number of selected attributes obtained by the corresponding classifier across the 12 databases are. The classifiers used in the experiments are: Naïve Bayes, c4.5, ibK(k=1), Support Vector Machines (LibSVM [13]), Multi Layer Perception (MLP), Tree Augmented Networks (TAN) and AODE; using for all of them the implementation provided by Weka [123]. Results for each database can be found in Appendix B.

When a cell is filled in as “–”, it means that the experiment could not finish after a (pre-fixed) very long execution time (commonly with Gisette or Dorothea databases).

Table 5.9: Results, for different classifiers, using IWSS^2 with CMIM-based re-ranking with block sizes B.

Classifier	**IWSS^2**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
NB	80.64	19.48	78.55	5.9	80.29	9.1	80.85	11.7	80.64	12.7	80.38	14.2	80.44	14.1
c4.5	80.70	13.8	79.32	5.4	80.34	6.8	81.08	8.4	81.36	9.2	80.87	9.8	80.94	10.6
ibK	85.01	19.9	82.01	7.0	82.53	9.5	82.71	12.7	83.91	13.7	84.42	14.0	84.40	15.2
SVM	71.28	7.19	70.82	2.8	71.69	3.5	71.33	4.1	71.42	4.4	71.43	4.3	71.62	4.4
MLP	–	–	81.23	6.7	81.51	7.6	81.48	8.6	81.37	9.4	81.13	9.3	81.75	10.5
TAN	82.73	17.9	81.39	6.7	82.38	9.0	82.55	12.1	83.43	13.3	84.10	13.7	84.15	14.4
AODE	85.76	17.36	84.96	7.1	86.23	9.3	86.47	12.5	86.21	12.2	86.22	13.4	86.26	13.8

Table 5.10: Results, for different classifiers, using IWSS$_r^2$ with CMIM-based re-ranking with block sizes B.

Classifier	**IWSS$_r^2$**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
NB	80.85	8.83	78.96	3.5	79.21	4.6	80.43	5.3	80.84	6.0	80.55	6.2	80.69	6.3
c4.5	80.16	9.1	79.21	3.7	79.73	4.4	80.26	5.3	80.45	5.7	80.76	5.9	80.84	6.4
ibK	82.05	8.8	82.79	4.9	81.89	6.3	82.76	7.3	82.24	8.6	82.65	8.9	82.69	9.5
SVM	–	–	70.95	2.3	71.82	2.9	71.22	3.3	70.61	3.4	70.85	3.5	70.79	3.5
MLP	–	–	80.51	4.2	80.10	4.8	80.95	5.6	82.25	5.8	81.74	6.1	81.44	6.1
TAN	–	–	81.09	4.2	82.37	5.8	82.60	7.0	83.73	7.8	84.34	8.4	–	–
AODE	86.64	9.5	84.31	4.4	85.57	5.8	85.92	6.2	86.34	7.3	85.93	7.1	85.54	7.9

Results shown in Tables 5.9 and 5.10 present one clear conclusion: the addition of the re-ranking criterion makes it possible to run algorithms IWSS^2 and IWSS$_r^2$ for all tested classifiers, while the original algorithms cannot be completed (in limited time) for all databases for complex classifiers (SVM, MLP and TAN).

Statistical tests are not run over the results since all cells are not completed, but it is easy to find a similar tendency to those in Tables 5.4 and 5.5: the larger the block size, the greater the accuracy and the size of the subset selected are, but statistically all of them seem equals to the original algorithm in terms of accuracy, but better in terms of size of selected subset. Regarding the classifiers' performance, it seems that AODE gets the best accuracy but pays for it by selected subsets of little more cardinality than most of the other classifiers.

5.4 Conclusions

This chapter has proposed a generic re-ranking algorithm which can be applied to incremental FSS algorithms. The idea behind re-ranking is that some features which are ranked at the end of the ranking used by the corresponding incremental FSS algorithm might become more relevant after the algorithm has selected some features from the beginning of the ranking. Thus, adding the re-ranking proposal to an incremental FSS algorithm lets these features come to earlier positions in the ranking, besides relegating features which are not relevant anymore after some feature has been selected. Furthermore, a stop-criterion has been proposed which perfectly suits the re-ranking proposal.

The re-ranking algorithm has been implemented with 3 different re-ranking filter criteria, 4 different FSS algorithms and 7 classifiers. Experiments and statistical tests prove that the re-ranking methodology drastically reduces the number of evaluations necessary, besides reducing the cardinality of the selected subsets while maintaining the accuracy obtained.

Chapter 6

Distribution-based Balance of Datasets in E-mail Foldering

6.1 Summary

A frequent problem found in databases coming from real life sources is the lack of balance among classes. There exist several methods to tackle this skewness, of whom this Chapter introduces a review and, besides, a new family of methods is presented: *Distribution-based* balancing algorithms, making comparisons and experiments using them and another state-of-the art method, on real life databases: ENROM e-mail foldering.

6.2 Introduction

This Chapter deals with databases representing e-mail documents. Thus, notation is changed respect to past chapters; thus, *instances* are also referred to as *documents* and consequently, *attributes* which form the instance vector are referred to as *terms*. Finally, the value of a term depends of the kind of representation of the documents: where the most common case is to represent documents by frequency; that is, the value of an attribute or term is the frequency such term appears in the corresponding instance or document.

Imbalance appears in a dataset when the proportion of documents/sub-concepts among classes/within-classes is very unequal. This has been a common problem in automatic classification but it has not been properly tackled until the recent appearance of highly skewed huge databases coming from real life sources (images, medical diagnosis, fraudulent operations, text. . . and some other corpora). In these databases the need for some preprocessing in order to alleviate the imbalance problem is a priority [17].

The degree of imbalance or skewness refers to the ratio among the number of documents from different classes. Thus, having a binomial class, a ratio of 1:100 would mean that the dataset contains 100 documents tagged with the majority class per each document tagged with the minority class. This problem is even worse when a class presents *absolute rarity* [118], an expression used to refer to the lack of data to properly learn a predictive model for such a class. From the literature we can learn that when dealing with skewed data, the major problem is not the imbalance itself, but the overlapping between classes or disjuncts [53; 87]. This problem is known as *between-class imbalance* [50] and is the type of imbalance dealt in experiments in this Chapter. Thus, experiments in Section 6.3 are expected not just to balance classes but also to remove between-class overlapping in the space region.

The imbalanced data problem can be approached from two different points of view: *algorithm-level* or *data-level.* Algorithm-level solutions are classifier-specific and consist in the introduction of a specific bias in the learned model [49; 63; 65; 91]. *Data-level* solutions are more popular and consist in the *a priori* modification of the training set [12; 16]. One way to do the former is by means of adjusting the degree of importance of each term [71] or just selecting some of them [130]. Alternatively, and this is the topic this Chapter focuses on, one can modify the training data in order to balance it, by sampling from the original dataset.

Sampling-based balancing techniques can be divided into *over-sampling* and *under-sampling*, although a combination of both can also be used. Besides this, sampling can be performed in a *directed* (intelligent) or *random* way. Over-sampling a training set consists in creating new samples (from minority class) and adding them to the training set, it being optional whether to remove the original samples or not. On the other hand, under-sampling chooses (in a random or directed way) samples belonging to the majority class and then removes them until the desired balance is achieved. Directed under-sampling is expected to remove documents of majority class(es) from regions which belong to minority class(es), while directed over-sampling is expected to reproduce more records of the minority class(es) and thus to define the region of that(those) class(es). Random under- and over-sampling only has the aim of balancing the training set, without taking care of removing important records. Sampling approaches were compared in [49] and the conclusions state that over-sampling and under-sampling perform roughly the same and, moreover, directed sampling did not significantly outperform random sampling. Later, in 2002 the well-known SMOTE algorithm was presented [16]. SMOTE is a combination of over and under-sampling whose application results in an improvement in the accuracy for the minority class. Anyway, the current situation is that no final word has yet been said about what approach is best [17].

Another aspect that must be taken into account is the fact that most of the approaches to the imbalanced data problem found in the literature refers to the problem of having a *binomial class* while the contribution in this Chapter faces with a *multi-class*, aggravated by the fact of having a large number of possible outcomes for the class variable. This point makes it difficult to transform the multi-class problem into a binary one [16], because the large number of (binomial) models to be learned heavily increases the time and space requirements of the process.

All the studies found in the literature which work on imbalanced multi-class datasets are very recent, for both *algorithm-level* [1; 131] and *data-level* [72; 110] solutions. At the moment, there are no clear statements about what imbalance solution performs best, and including the multi-class paradigm adds a new complexity level. However, one may find that this case is more realistic when the problem tackled is related to text categorization, as this Chapter aims.

6.3 Methodological Contributions: Distribution-based balancing of multi-class training sets

The approach presented here to deal with imbalanced data is based on a two step process: (1) for each predictive attribute or term t_i, $i = 1, \ldots, r$ and for each class state c_j, learn a probability distribution $P(t_i|c_j)$; and (2) for each class state c_j, sample b full instances[1] $\langle f_1, \ldots, f_r, c_j \rangle$ by using the r previously learnt probability distributions $P(t_i|c_j)$, $i = 1, \ldots, r$. It is clear that if we sample the same number of instances for each class state, then we get an *artificially generated balanced* training set. Therefore,

[1]Notice that a full instance contains the frequency f_i of term t_i in the sampled document plus the class the document (e-mail) belongs to.

taking into account the concepts introduced in the previous section, our method belongs to the category of *data-based* algorithms that combine *under-* and *over-sampling* with total replacement of the training set.

As mentioned in Section 6.2, the problems when dealing with imbalanced data come not only from the fact of having skewed data but also from between-class overlapping. It is expected that in many cases the contribution presented (*Distribution-based balancing of multi-class datasets*) can manage these two problems simultaneously. The idea is that when learning $P(t_i|C = c_j)$ we are trying to represent class state c_j by term t_i. If c_j is a majority class state, then when the distribution is sampled, the probability of producing outliers is small, and so, the probability of invading other class states also decreases. On the other hand, if c_j is a minority class, as we learn its concept for each term independently, the noise coming from other class states is removed, and because we propose to sample the same number of *artificial* instances for each class label, then its corresponding concept will be more clearly defined with respect to majority classes. In this reasoning, it is assumed that no disjunction exists, that is, that, in general, a class state does not cover different sub-concepts and so the concept can be represented by a uni-modal distribution. If disjunctions exists, overlapping among classes cannot be removed and so our proposal will correct the skewed problem but not between-class overlapping. However, assumption is that the existence of class states corresponding to different sub-concepts is more likely to exist in binomial problems, while it is not so frequent in multi-class problems. Experimental results confirm this expectation.

In short, this proposal performs several tasks over the training set in a single process:

- Over-samples classes with less than b documents.
- Under-samples classes with more than b documents.
- Reduces over-lapping among classes.
- Fully balances all classes.

Figure 6.1 shows the general scheme of the proposed distribution-based algorithm. As it can observed there are two clearly differentiated steps: learning and sampling. At learning time a probability distribution for each pair ⟨term,class state⟩ is learnt from the *unmodified* training set (corresponding to the current split of the validation) projected over the term and the class ($D_h^{\downarrow t_i,C}$). Then, at sampling time all the distributions learnt for a class state are used to build (by sampling the corresponding one for each term in turn) each one of the b desired documents for such a class state. This sampling process is repeated for all the class states in order to get a balanced *artificial* training set.

```
In    $D_h$ training set, $C$ class variable
      $|V| = r$ the number of terms/features, $b$ #new_instances to sample per class,
Out   $newD_h$ whole new and fully balanced training set
      /* learning phase */
1     for each class $c_k \in C$ do
2       for each term/feature $t_i$, $i = 1, \ldots, r$ do
3         learn probability distribution $P_{ik}$ from $D_h^{\downarrow t_i,C}$
      /* sampling phase */
4     $newD_h \leftarrow \emptyset$
5     for each class $c_k \in C$ do
6       for p=1 to $b$ do
7         newDoc = new double[r+1]
8         for each term/feature $t_i$, $i = 1, \ldots, r$ do
9           newDoc[i] = sample value from $P_{ik}$
10        newDoc[r+1]=$c_k$ //add class label
11        $newD_h = newD_h \cup newDoc$
12    return $newD_h$
```

Figure 6.1: Distribution-based balancing algorithm.

Of course there are some degrees of freedom in the previous algorithm: the number of documents to be sampled for each class and (mainly) the kind of probability distribution used to model the training set. Experiments for this contribution are instanced with four different probability distributions: *Uniform, Gaussian, Poisson* and *Multinomial*.

- Uniform Distribution. There are several works in the literature (e.g. [49]) which, using binomial classes, conclude that there is not much difference whether sampling by using information extracted from the learning data, or not. Sampling (almost) without using information about the training set can be modeled by learning a uniform distribution. In this way the only information we collect is the *max* value found for term t_i restricted to those samples belonging to class

c_k. Later, in the sampling process an uniform number is generated in the interval $[0, max_{ki}]$. This distribution is used as a baseline threshold to analyze the advantages of the more informed ones.

- Gaussian Distribution.

  The univariate Gaussian distribution assumes the frequency f_j of term t_i conditioned to class state c_k as follows:

  $$f(t_i = f_j) = \frac{1}{\sigma\sqrt{2\pi}} exp\left[-\frac{(f_j - \mu)^2}{2\sigma^2}\right].$$

  Thus, learning a univariate Gaussian Distribution consists simply of computing the mean and standard deviation of frequencies for term t_i from data restricted to class state c_k. Sampling from a Gaussian distribution can be done, for example, following the well known Box and Muller method (see for example [94], chapter 2).

  It should be pointed out that whenever the sampled value is less than 0, it is set to 0 since we are working with frequencies and negative values make no sense for the case of work (text documents).

- Poisson Distribution. As pointed out in [58] “if we think that the occurrence of each term is a random occurrence in a fixed unit of space (i.e. a length of document), then the Poisson distribution is intuitively suitable to model the term frequencies in a given document”. Because of this, the Poisson model has been investigated in the information retrieval community and applied to text classification [58]. Thus, it is expected to be a good alternative to be considered for distribution modeling in our balancing algorithm.

  The Poisson distribution assumes the frequency f_j of term t_i as follows:

  $$P(t_i = f_j) = \frac{e^{-\lambda}\lambda^{f_j}}{f_j!} \tag{6.1}$$

  where λ is the mean.

  Therefore the learning step is simply a matter of computing λ of term t_i restricted to class state c_k. Sampling from a Poisson distribution is also a well-known process (see for example [94], chapter 2).

- Multinomial Distribution.

  Following the generative distribution from the Naive Bayes Multinomial Model [57] described in formula 6.2[1], and once we have learnt the term distribution by class following expression 6.3, we are ready to generate as many documents as the parameter b indicates.

$$P(d_i|c_j) = P(|d_i|)|d_i|! \prod_{l=1}^{r} \frac{P(t_l|c_j)^{M_{il}}}{M_{il}!} \tag{6.2}$$

$$P(t_l|c_j) = \frac{1 + \sum_{i=1}^{|D|} M_{il} P(c_j|d_i)}{|V| + \sum_{s=1}^{r} \sum_{i=1}^{|D|} M_{is} P(c_j|d_i)} \tag{6.3}$$

  M_{il} being the number of times that term t_l appears in document d_i, $r = |V|$ the size of our vocabulary, $|D| = m$ the number of documents to classify and $|d_i|$ the length of document i. Notice that $P(c_j|d_i)$ in equation 6.3 is simply 1 if the instance corresponding to document d_j is labeled with class c_j in the dataset, and 0 otherwise. Equation 6.2 assumes independence among terms, which is not realistic in real databases. Besides, this assumption gets even more troublesome in the multinomial model [64] because it assumes not just independence among different terms but also among various occurrences of the same term.

  In this case the $P(|d_i|)$ distribution is assumed to follow a Poisson distribution, Equation 6.1, which is unidimensional and independent of the class. So by the estimation of the parameter λ as the mean number of terms in a document, that is, the mean length of documents, we can simulate the number of terms in a newly-generated document. Once the number of terms is given, the terms are picked in the generated document by simulating the probabilities $P(t_l|c_j)$ as many times as the number of terms has been indicated by the Poisson distribution.

  So, in the structure of the algorithm shown in Figure 6.1, the changes are:

  (1) Values to compute are λ and the probabilities for each term given the class following expression 6.3.

[1]Where $|d_i|$ stands for number of words in document d_i; $P(t_l|c_j)$ stands for the probability of finding term l having frequency t_l in a document belonging to class c_j

(2) Each term will have the value representing the times that it was *drawn* following the Multinomial distribution $P(t_l|c_j)$ according to the length of document generated following a Poisson distribution.

6.4 Experiments on Text Categorization

This section presents an experimental study over e-mail foldering using the proposed distribution-based balancing method. It should be emphasized that this experiment is directly related with e-mail foldering and because of the nature of this problem, perhaps the conclusions here obtained can be extended to similar problems, that is, those having a class variable with large cardinality and numerical variables as predictive attributes. For the sake of completeness, apart from using our approach instantiated with Uniform, Gaussian, Poisson and Multinomial distributions, it is also considered the well-known SMOTE algorithm, but slightly modified to deal with multi-class datasets.

6.4.1 Obtaining datasets: from text e-mails to a structured dataset

The main differences between standard classification and text classification are: the need for preprocessing the unstructured documents in order to obtain a standard data mining dataset (bi-dimensional table) and the usually large number of features or attributes in the resulting dataset. Two other important differences with respect to standard text classification tasks are the large number of states in the class variable, and the usual presence of *noise* in the training set, due to the fact of (almost) all users of e-mail, even having defined topical folders, later tend to file e-mails belonging to different concepts into the same folder. This experiments focuses on the *bag-of-words* model, that is, a document (mail) is regarded as a set of words or terms without any kind of structure. The selection of the documents and terms (i.e., the vocabulary V) used in this study follows the preprocessing described in [8]:

- Documents: Non-topical folders (inbox, sents, trash, etc.) and folders with only one or two mails are not considered.
- Terms: Only consider words as predictive attributes (MIME attachments are removed) and no distinction is made with respect to where the word appears (e-

mail header or body). Stop-words and words appearing only once are removed. After that we denote the size of V as $|V| = r$.

- Class: The folder hierarchy is flattened and each one of the resulting folders constitutes a class label or state.

The representation of documents is also an important issue. The most typical representations are *frequencies* and *tf-idf*. The former represents a document using a vector which contains the frequencies in that document of terms belonging to a predefined bag-of-words or vocabulary. The latter also uses a vector, but in this case the position of each term represents a mix of the frequency in that document and its frequencies in the rest of the documents. Other not so usual representations are *n-grams* [7], *hypernyms* [103], entities [122], etc. The current literature is not able to say which representation performs best, so the decision still depends on the case under study and the type of input accepted by the classifier used.

For the sake of completeness, in this experiment two different kinds of representations are used for the dataset. In particular, the vectorial model is used to represent documents, thus, after using information retrieval techniques [101] to carry out the previously described preprocessing, the datasets can be observed as bi-dimensional matrices $M[\text{numDocs}, \text{numTerms}]$, where $M[i, j] = M_{ij}$ is a real number representing (1) the frequency of appearance of term t_j in document d_i or (2) the tf*idf values normalized by the cosine function.

6.4.2 Test suite

As in [8; 61], datasets used correspond to seven users from the ENRON corpus (mail from these users and a temporal line in increasing order can be downloaded from http://www.cs.umass.edu/∼ronb). The downloaded data was preprocessed according to the process described in Section 6.4.1. To do this it was necessary to code a home-made program in Java, and design it to interact with *Lucene* information retrieval API[1] and to output a sparse matrix $M[\text{numDocs}, \text{numTerms}]$ codified as an *.arff* file, i.e., a file following the input format for the WEKA data mining suite [123].

Table 6.1 describes the main characteristics of the datasets obtained. The last two columns are intended to show the degree of imbalance in the datasets. As in [110],

[1]http://lucene.apache.org/who.html

(*base*,*peak*) is shown, where *base* is the number of documents of the minority class and *peak* is the number of documents in the majority class. However, it is expected that this measure is not sufficiently representative of the degree of imbalance because it represents different distributions in the same way, e.g., (1,100) is valid for $\{1, 100\}$ or $\{1, 100, 100, 100\}$ or $\{1, 100, 1, 100, 1, 100\}$, which clearly represents different degrees of imbalance. Because of this a new feature is added to the description: (μ,σ), μ being the mean of documents per class and σ the standard deviation of the mean. σ is found to be a very informative value concerning the imbalance present in the dataset. A clear indication of the degree of imbalance in the problem dealt (e-mail foldering) is that in the seven users (datasets) considered, the standard deviation is greater than the average. Besides this, it is also computed the kurtosis degree for each dataset, which in this case represents the degree of concentration of the number of documents per class around the mean number of documents per class. A graphical representation of the imbalance in datasets is shown in a Box and Whisker plots (Figure 6.2), where the inputs used were the number of instances per class state.

Table 6.1: Instances, Classes, Attributes, and degree of imbalance in the datasets

	#I	#C	#A.	$(base : peak)$	$(\mu \pm \sigma)$	Kurtosis
lokay-m	2489	11	18903	(6:1159)	226.3±316.3	2.75
sanders-r	1188	30	14463	(4:420)	39.6±75.6	17.05
beck-s	1971	101	13856	(3:166)	19.5±24.5	13.08
williams-w3	2769	18	10799	(3:1398)	153.8±379.4	4.10
farmer-d	3672	25	18525	(5:1192)	146.9±255.5	7.77
kaminski-v	4477	41	25307	(3:547)	109.2±141.9	1.66
kitchen-l	4015	47	34762	(5:715)	85.4±122.8	11.75

6.4.3 Experimental design

The goal is to study whether the classifiers, specially Naive Bayes Multinomial (which is designed for frequencies representation), perform better after balancing the datasets. To be more confident in our conclusions, a statistical analysis is carried out in which we compare the results with and without balancing, and a contrast is also made with the results obtained using different classifiers. The experiments deal with the following *actors*:

- Balancing algorithms. To balance the training sets the proposed method is instantiated with the four mentioned distributions: Uniform, Gaussian, Poisson

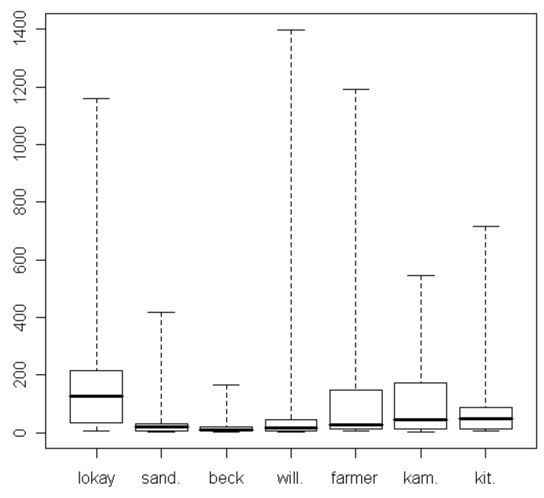

Figure 6.2: Graphical representation of the imbalance degree in the seven e-mail users.

and Multinomial. For comparison, it is also considered SMOTE [16]. SMOTE was initially proposed by its author for binomial classes and performs the balancing by sampling synthetic documents from a minority class in a given percentage, and then randomly under-sampling as many majority class documents as desired. To apply SMOTE in a multi-class problem, it is necessary to select the number (b) of documents to re-sample for each class and apply SMOTE to over-sample, without replacement (the SMOTE algorithm does not perform replacement), all classes with cardinality lower than b until obtaining b documents for those classes; then, for classes with cardinality over b, there is to randomly remove as many documents as necessary to get cardinality b.

- Document representation. Documents have been represented by using frequencies and tf*idf. Computation of tf*idf is done incrementally, that is, trying to reproduce the fact that we have a dynamic problem and not a static one, where tf*idf can be computed at the beginning by using all the available information. In this case the classifier should categorize incoming e-mails as they are downloaded into the user inbox folder. So, when testing using time-based split evaluation (see Section2.2.1), test e-mail i should be represented with a tf*idf value where the idf part is computed using the training set and test e-mails from 0 to $i-1$, instead of using all s test documents, since documents $i+1$ to s are not supposed to be in the inbox folder yet. In this way, incremental time-based split validation needs more time but is more realistic.

When balancing is carried out by learning any of the distributions presented in Section 6.3, the parameters needed $(max, \mu, \sigma, \lambda)$ are computed using frequency representation, and then the new training set is sampled still using frequencies representation. Thus, tf*idf conversions are performed later when training and testing the classifier.

- Classifiers. The three different classifiers used are commonly considered when dealing with text categorization problems: Naive Bayes Multinomial (NBM) [78], Support Vector Machine (SVM) [10] and *k*-Nearest Neighbors (*k*-NN) [30]. For NBM it was used the implementation provided in the WEKA API. For SVM, it was used the implementation available in WLSVM [25], which can be viewed as an implementation of the LibSVM [13] running under WEKA environment. For *k*-NN, an own implementation was coded by using WEKA API. Different values of *k* were tested and two distance metrics: Euclidean and Cosine distances. Only the results for the best configuration found are shown: *k*=15 and the Cosine distance which is also the standard similarity metric in text, and it is computed by: $sim(u,v) = \frac{\vec{u}\cdot\vec{v}}{|\vec{u}|\times|\vec{v}|} = \frac{\sum_{i=1}^{n} f_{iu}\times f_{iv}}{\sqrt{\sum_{i=1}^{n} f_{iu}^2 \times \sum_{i=1}^{n} f_{iv}^2}}$ Where *n* is the number of dimensions (attributes) of each vector and f_{iu} the value of dimension *i* in vector (document) *u*.

Having the previous setting in mind, three different experiments are to be run:
Experiment 1.- To compute the baseline against which to compare, an incremental time-based split evaluation is ran on each user in Table 6.1 without balancing. With respect to the value of s in the time-based split evaluation, in all the experiments carried out it has been used the value recommended in Bekkerman's work, that is, the number (s) of e-mails to classify in each split is 100. The three mentioned classifiers were ran for both types of document representation: frequencies and tf*idf. SVM performed statistically the same in both cases so from now on it is only shown its results for tf*idf, which is the configuration suggested in the literature. For k-NN, only results are shown for tf*idf because that is the configuration which finally obtains the best results. Finally,for NBM results are shown for both representations: frequencies and

tf*idf. The results are tabbed in Table 6.2, where each entry represent the accuracy averaged over all the folds tested in the time-split validation process.

Table 6.2: Baseline accuracy for 7 e-mail users.

	NBM freq	**NBM tfidf**	**SVM tfidf**	**k-NN tfidf**
lokay-m	75.27	63.65	79.12	39.54
sanders-r	55.51	48.90	66.20	35.97
beck-s	28.26	17.60	36.87	9.00
williams-w3	91.69	88.87	90.52	86.06
farmer-d	69.64	55.34	73.98	37.63
kaminski-v	45.61	34.75	54.26	9.92
kitchen-l	32.19	34.44	51.10	14.28
Mean	**56.88**	**49.08**	**64.58**	**33.20**

Experiment 2.- In this experiment a test is made for the effects of our proposal, that is, instead of learning the classifiers from the original dataset, they are learnt from the artificially balanced dataset. Because of the randomness introduced by the sampling process,each balancing algorithm is ran five times and show averaged results. In all the cases the number of documents b to sample per class is set to 30. Tables 6.3 and 6.4 show the results obtained. Each entry in the tables represents the accuracy averaged over all the folds tested in the time-split validation and coming from the five independent executions. Comparison between baseline algorithms (Table 6.2) and balancing algorithms was done by using a Wilcoxon signed rank test [23; 121] (α=0.05), taking as input the time-split values from the corresponding user-classifier. When the balanced user-classifier is found to be statistically better than the baseline (not balanced), a ● is placed inside its corresponding cell.

Table 6.3: Results when balancing with the SMOTE algorithm.

	NBM freq	**NBM tfidf**	**SVM tfidf**	**k-NN tfidf**
lokay-m	67.94	●67.72	68.82	●57.79
sanders-r	●72.21	●73.28	59.59	●69.82
beck-s	●45.41	●45.49	●42.80	●39.86
williams-w3	87.20	74.43	87.75	64.84
farmer-d	61.21	44.70	72.77	37.15
kaminski-v	43.22	●42.42	49.65	●34.85
kitchen-l	35.92	●38.54	49.72	●32.68
Mean	**59.02**	**55.23**	**61.59**	**48.14**

Table 6.5 shows a paired comparison between each two kinds of balancing methods over the four classification methods used (NBM freq, NBM tdidf, SVM tfidf and k-NN tfidf). Comparison is in the form $x - y - z$, where x stands for $\#beats$, y stands for

Table 6.4: Results when balancing with the proposed distribution-based algorithm.

	NBM freq	**NBM tfidf**	**SVM tfidf**	**k-nn tfidf**
lokay-m	52.02	59.20	36.70	49.24
sanders-r	59.33	68.78	71.99	66.60
beck-s	●43.32	●48.76	39.93	●43.13
williams-w3	63.56	66.44	46.42	38.81
farmer-d	52.05	51.38	35.29	●46.31
kaminski-v	40.69	●51.80	28.72	●49.52
kitchen-l	33.11	●42.05	26.11	●37.18
Mean	**49.15**	**55.48**	**40.74**	**47.26**

(a) Uniform Distribution

	NBM freq	**NBM tfidf**	**SVM tfidf**	**k-nn tfidf**
lokay-m	70.77	●71.51	47.58	●55.95
sanders-r	72.39	●75.23	74.79	●76.02
beck-s	●45.88	●50.55	●44.33	●45.00
williams-w3	84.27	80.78	81.50	77.22
farmer-d	67.11	●64.15	65.93	●54.41
kaminski-v	●52.02	●56.68	55.39	●52.20
kitchen-l	●38.72	●46.45	37.24	●34.66
Mean	**61.59**	**63.62**	**58.11**	**56.49**

(b) Gaussian Distribution

	NBM freq	**NBM tfidf**	**SVM tfidf**	**k-nn tfidf**
lokay-m	70.68	●73.20	69.95	●60.35
sanders-r	●75.40	●75.88	57.09	●74.09
beck-s	●45.17	●48.53	41.17	●44.05
williams-w3	88.93	77.96	85.62	66.65
farmer-d	65.87	55.25	68.28	●48.72
kaminski-v	46.54	●49.36	45.67	●45.07
kitchen-l	●37.10	●41.92	39.95	●38.44
Mean	**61.39**	**60.30**	**58.25**	**53.91**

(c) Poisson Distribution

	NBM freq	**NBM tfidf**	**SVM tfidf**	**k-nn tfidf**
lokay-m	70.01	●72.41	67.94	●61.25
sanders-r	74.97	●74.84	55.30	●71.66
beck-s	●45.09	●49.33	40.48	●46.19
williams-w3	89.03	78.86	85.47	66.72
farmer-d	63.22	54.49	70.25	●47.55
kaminski-v	44.48	●47.49	46.10	●42.34
kitchen-l	●35.84	●41.57	40.00	●37.34
Mean	**60.38**	**59.86**	**57.93**	**53.29**

(d) Multinomial Distribution

$\#ties$ and z stands for $\#loses$. For example, in the first cell, the comparison 15-5-8 means that the SMOTE method performed statistically better than the Uniform Distribution method 15 times, statistically the same 5 times and has performed statistically worse 8 times.

Table 6.5: Paired comparison between balancing methods.

	Uniform	Gaussian	Poisson	Multinomial
SMOTE	15-5-8	6-7-15	4-7-17	4-11-13
Uniform		2-3-23	3-8-17	3-10-15
Gaussian			13-11-4	14-10-4
Poisson				10-16-2

Experiment 3.- Finally a study is carried on to measure the effect of b on the performance of the distribution-based balancing algorithms. The focus is on a representative case as is the case of using the Gaussian distribution for balancing, the NBM classifier and tf*idf document representation. This configuration was ran using values for *b* from 10 to 60, and the results are shown in Figure 6.3.

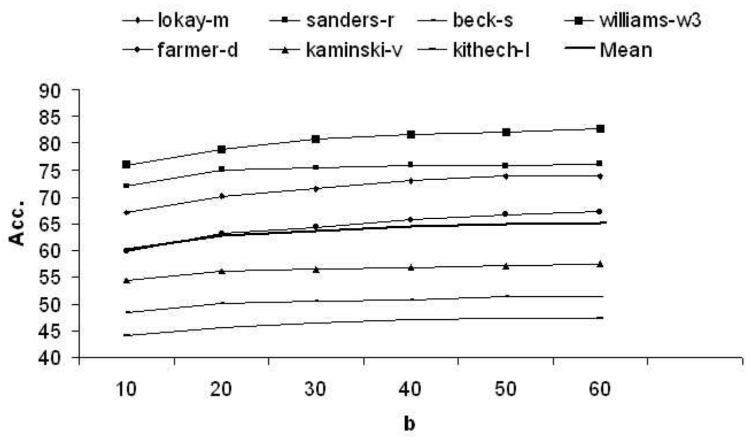

Figure 6.3: Different values for *b* using NBM and tf*idf docs. representation.

6.4.4 Discussion of Results

Experiment 1: baseline. Table 6.2 shows the results obtained after performing time-based split evaluation on seven users from the Enron Corpus without preprocessing that instances set. Results show, as in [8], that SVM-tfidf outperforms by far other common classifiers such as NBM. Users with worst results are those with a lower σ value in Table 6.1, which also corresponds with the higher cardinality of class. Thus, it is found in this dataset that high class cardinality leads to low standard deviation in the number of documents per class, and this is an indicator of classifiers' performance on such databases. We can also see that *Williams* gets a very high performance so it is not an easy target to improve.

Experiment 2: balancing Table 6.3 presents the results obtained when balancing training sets using the *SMOTE* method, and Table 6.4 shows results for the proposed distribution-based balancing methods. Both tables include a • symbol when the algorithm associated to a cell performs statistically better than its corresponding cell in the baseline in Table 6.2. Finally, a pairwise comparison between balancing methods is presented in Table 6.5.

SMOTE balancing proves to be a good choice for preprocessing skewed datasets. Its results outperform the baseline for some users, specially for NBM-tfidf and k-NN-tfidf classifier configuration. On the other hand, SVM only gets statistical improvement in one user and it even decreases in some others.

The proposed distribution-based methods can be classified as one random (Uniform) and three directed (Gaussian, Poisson and Multinomial). Results for Uniform re-sampling are the worst although several statistical improvements are achieved. Comparing Gaussian, Poisson and Multinomial re-samplings by looking at Table 6.5 conclusion is that a reasonable order from best to worst could be: $Gaussian > Poisson > Multinomial > Uniform$. In particular, the best choice seems to be Gaussian balancing with the NBM classifier and tf*idf representation. Furthermore, as happens when using *SMOTE*, the SVM classifier does not get real improvement after balancing, thus indicating that SVM is strong in imbalanced situations, so performing re-sampling, at least in the case of using our methods and *SMOTE*, does not provide any improvement and even worsens the learning stage. This corroborates a study by [51] which concludes that SVM is robust against imbalanced datasets. Finally, comparing random and directed re-sampling, it is found support for the personal hypothesis which expected that directed balancing alleviates the overlapping problem. Moreover, this provides evidence to suggest that imbalance is not the only problem in skewed datasets.

When comparing *SMOTE* against the proposed distribution-based methods in Table 6.5, it is found that *SMOTE* performs better than the random distribution-based method, but worse than our three suggestions for directed distribution-based methods. This clearly concludes that the methodological proposal for balancing (except for Uniform distribution) outperforms *SMOTE* at least when applied to text categorization with more than 2 classes.

With respect to document representation, if we focus on NBM, whose results are shown with frequencies and tf*idf documents representation, we can see that in the baseline NBM with frequencies performs better than NBM with tf*idf but, after balancing the data, NBM with tf*idf performs considerably better and achieves statistically the same results as NBM with frequencies; thus, this can be interpreted as empirical proof that tf*idf representation is more imbalanced-sensitive than frequency representation.

Comparing users by looking at Table 6.1, Figure 6.2 and Table 6.4, users which usually obtain statistical improvement are: beck-s, kaminski-v, kitchen-l and sanders-r. These four users are those with a larger cardinality for their class attribute and which present lower outliers in their box and whisker plot. Furthermore, with respect to their (μ, σ) representation, they are also the user with lowest μ and σ values; respect to kurtosis degree, higher values clearly point to a higher need of balancing. Thus, based on these results, it is suggested that a good way to predict performance improvement after balancing a dataset may be one of these: (1) cardinality class, (2) outliers in box and whisker plot, (3) low (μ, σ) values and/or (4) high kurtosis values . For example, by looking at Table 6.1 we could predict that a good order of datasets with a greater need for a balancing process are (from more to less): beck-s, sanders-r, kitchen-l, kaminski-v, farmer-d, lokay-m and williams-w3. Finally, one may suggest using the kurtosis value since this is just a single value and very related to the balancing need.

Experiment 3: number b of documents per class. The configuration Gaussian Distribution with classifier NBM and tf*idf representation was ran using values for *b* from 10 to 60, and the results are shown in Figure 6.3. Choosing the value for *b* should not only be based on performance but also on the computational cost of sampling b new instances for each class. Thus, based on Figure 6.3, it is suggested using values from 30 to 40, since the computational cost is quite high from 30 documents and above, while the improvement achieved is not significant.

6.4.5 Conclusions for Distribution-based balancing applied to Text Categorization.

For the NBM classifier, a great improvement on its performance was achieved for the e-mail foldering task, what is of great interest given that NBM is a well-known standard for text classification.

A comparison has been presented of four kinds of distributions (one random and three directed) to fully re-sample training datasets for multi-class classification, applied to e-mail categorization; besides, another comparison was made for the proposed contribution results and the well known *SMOTE* balancing method. The results support the main hypothesis, which stated that a directed re-sampling method not only balances the training set but also reduces the problem of overlapping among classes, while random re-sampling is only capable of dealing with the imbalance problem. Furthermore, evidence is found that the SVM classifier is very robust under imbalanced conditions, so its performance does not improve after balancing. Comparing with *SMOTE*, the presented distribution-based methods statistically outperform it, except for the Uniform distribution.

Finally, common characteristics have been found between datasets which usually perform better after balancing; that is, it can be expected that datasets whose performance improves after balancing will be those which present a large cardinality class, low outliers in a box and whisker plot, low (μ, σ) values and/o high kurtosis degree.

6.5 Conclusions

The problem of class imbalance in datasets has been presented in this Chapter. A new kind of balancing algorithms has been proposed: *Distribution-based* balancing, and they have been compared among themselves and with the well-known SMOTE algorithm; besides, experiment have been ran on a Text Categorization problem (E-mail foldering). The main conclusion obtained is that the proposed Distribution-based methods outperform SMOTE in terms of Accuracy, concretely the multinomial model performs quite well and thus it is an interesting contribution to the e-mail foldering problem.

Part III

Preprocessing Algorithms in Multimedia Databases

Chapter 7

Video Representation in MIR

7.1 Summary

In this chapter an introduction is presented to Multimedia Information Retrieval, focusing on the "Semantic Gap" problem and the representation of multimedia documents, mainly "Visual Features". Finally, several experiments are reported about visual features selection, dimensionality reduction and about other complementary kinds of representation for multimedia documents.

7.2 Introduction to Multimedia Information Retrieval

Multimedia Information Retrieval (MIR) [4] is a relatively new subfield of the Information Retrieval (IR) problem. Usually, when one refers to IR it is automatically assumed that one means "text" IR (TIR). However, the last two decades have seen the birth of a new discipline, MIR, which has the same goal (suggestion of relevant documents given a search topic) but applied to a wide variety of multimedia documents which may contain: (hyper)text, video, sound and/or still images. With the exponential growth of the Internet and multimedia contents, the MIR discipline has attracted the attention of the research community over the last years.

An IR system presents the components stated in Definition 7.1:

Definition 7.1 *An **Information Retrieval model** is a sixtuple: [**D**,$\{$**Q**,**C**$\}$, O, $\{$R(q_i,d_j), R'(c_i,d_j)$\}$*] where:

1. ***D*** *is the set of stored documents*
2. ***Q*** *is the set of queries a user can perform*
3. ***C*** *is the set of categorizations the user may request*
4. *O is the set of operations that can be applied on Documents or Queries*
5. *R(q_i,d_j) is a function returning a real value. This value is used to insert document d_j in a ranking of relevant documents for query q_i.*
6. *R'(c_i,d_j) is a function returning a real value. This value is used to insert document d_j in the corresponding set in order to perform the categorization pointed out in c_i.*

And we can find a more natural definition in [75]:

Definition 7.2 ***Information Retrieval*** *is finding documents of an unstructured nature, usually text, that meet an information need from within large collections, usually stored on computers.*

Multimedia data has become as much a part of our daily lives as text documents. However, retrieval of the former presents much more troublesome issues than the latter. In the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval, the Panel Session [48] discussed the main problems in MIR:

- Semantic Gap. Multimedia documents are usually described using *low-level features* (see Section 7.4) such as *Color Layout* or *Texture*. The semantic gap refers to the difficulty of expressing or representing high-level concepts using these kinds of features. This is not so problematic in text IR, where the semantics of text is (almost) linked to the content of the document.

- Query-type. We can find multimedia queries performed by text (here the semantic gap is most important) and by sample (e.g.: search for similar images to the one provided as input). Both techniques lead to different search systems and possibilities.

- Scalability and Search Speed. Storing and Processing multimedia documents requires low search-complexity MIR algorithms.

- Benchmarking. Public and common evaluation databases are required to demonstrate that MIR techniques are profitable in general and not just in particular studies. One of the most important movements in this aspect is being made by the TRECVID[1] [107] research effort, which aims to tackle the most difficult problems of content-based video retrieval.

- Modeling. "Move away from a *one model fits all* approach" is a key decision, which would lead to retrieval systems which would use different algorithms or document representation depending on factors such as the search topic or user context. For example, voice transcription could be most important in news retrieval while it would not be useful for unlabeled images collections.

- User Interfaces and Human Interaction. The difficulty of performing multimedia retrieval might be alleviated by designing interfaces [4] which help the user to perform better queries (text and content-based) orwhich return documents in a more user-friendly manner.

7.3 Semantic Gap

There is a certain agreement in the research community that the Semantic Gap is the most important problem in MIR engines. The unsatisfactory performance of video retrieval systems is mainly due to the gap between low-level description (see Section 7.4) of multimedia documents and their high level semantics.

In [108] we find this definition:

Definition 7.3 *The semantic gap is the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation.*

How to link the physical descriptions of a document to the understanding of a human being is part of the IR subfield known as *Content-based Information Retrieval* (CBIR) [108] and, up to now, the semantic gap has not been satisfactorily bridged. The results of TRECVid experiments up to now show the inadequency of content based search systems and also the limited effectiveness of high-level feature extraction systems.

[1]TRECVid is a large-scale evaluation campaign aimed at research problems related with video data.

One straightforward approach to solving this problem is the manual annotation, description or indexing of multimedia documents, which reduces the content-based problem to the original task of (text) information retrieval. When applying these labeling techniques, two main problems arise identified: (1) *Cost*, since it is clear that the manual description of thousands of images is extremely expensive in time and staff, although some automatic solutions haven been adopted [14; 96]; and (2) *Coverage*, which aims that the whole semantics of an image cannot be captured by text description [3]. So manual labeling is not expected to solve the semantic gap problem in all cases and so CBIR is nowadays a very active research field.

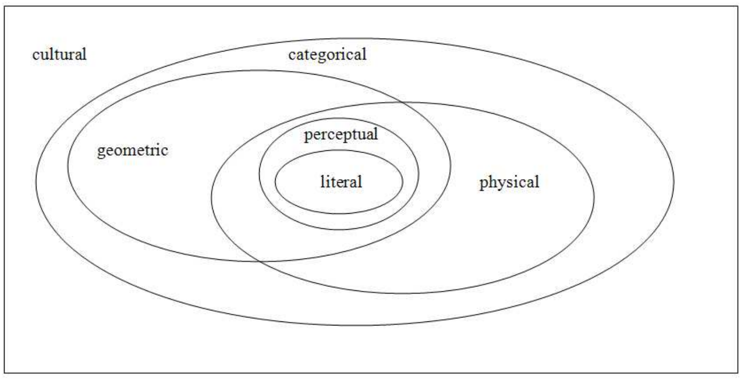

Figure 7.1: Classes of Equalities for Content-Based Image Search.

Figure 7.1 (adopted from [108]) shows all the semantic equalities that a CBIR system needs to address. If we are searching for any kind of “car”, a categorical equality is enough. However, we might aim for more complex requirements as: blue car (*physical*), long car (*geometric*) or nice car (*perceptual*). The most restrictive equality between images is “literal”, which would seek for literally identical objects. Thus, bridging the semantic gap would mean building a CBIR system which can link the semantic similarity a user is looking for and the data processing of multimedia documents that computers provide.

There are some other fields of study to improve the performance of MIR systems, such as the design of collaborative systems [43] which take into account the implicit and/or explicit feedback of other users; the design of more powerful multimedia search interfaces [2; 114] which help guide the user in his/her specific needs; and taking into account the context of the user [45]. For example, the retrieval systems evaluated in the TRECVid model retrieval in a “one result list only” approach, which assumes the user is focused on one particular search issue. An example of this type of search task is: *“Find shots of Condoleezza Rice”*.

These tasks are useful in benchmarking various retrieval algorithms as shown in the TRECVid evaluation experiments, but they are not representative of real world video information seeking tasks. For example, a researcher or journalist at a broadcasting station who is searching for material to use in the production of an item for the evening news, may be interested in highlighting the achievements of multiple swimmers at the 2008 Olympic Games in Beijing. However, as they progress through the search task, they may become interested in highlighting other issues, such as preparatory issues related to the performance of Michael Phelps, or to highlight the need for more governmental support in the development of future swimmers. Current retrieval systems and approaches fail to provide any support for such broad, multi-faceted tasks. In a faceted retrieval system, one may search for information about various aspects of the underlying information need without interrupting the current search session. These kinds of interfaces might be regarded as supportive complements, but the main problem in CBIR is still found in the representation and processing of multimedia documents. So this part of the thesis focuses on the representation of still images (video key-frames) to help their processing and relevance prediction. Besides, in Chapter 8 a deep study of the influence of context on classifiers is presented.

7.4 Visual Features

Visual Features [108] is another way of referring to *low-level features* or *object features*. These are arrays of real values describing a physical property of an image. As mentioned in Section 7.3, the use of visual features is a key factor in CBIR for MIR systems.

7.4.1 Visual Features Description

One of the main goals in CBIR is the automatic indexing of multimedia documents such as images and videos. The information stored for each document must be as self-describing as possible and compatible with relevance prediction algorithms which need to compute the high-level semantics of the document.

Visual features are used to describe both still images and video shots[1]. In order to describe a video shot, several key-frames are (automatically) selected to represent such a shot and then those key-frames are represented using the same features as for still images.

To extract visual features from images or video shots, several tools can be used, the most-used currently being the MPEG-7 Visual Standard for Content Description [106]. Text created from transcript speech is another common way of representing shots. As stated in [125], although it can be used to gain good performance, it cannot be applied to all videos in general due to the lack of speech in some videos, or the fact that the speech does not relate to the visual content of the video.

Some of the most commonly used low-level features are:

1. **Color Histogram**. There exist several color spaces (RGB, YCbCr, HSV) which have a more discriminanting power than grey scale color descriptions. The Color Histogram [111] in an image is one of the color descriptions most commonly used, and it is invariant to translation and rotation. The MPEG-7 standard codifies this feature in the HSV color space, which makes the hue property invariant with respect to the illumination.

2. **Color Layout** [74]. This vector is the result of a Discrete Cosine Transformation (DCT) over a 2D array of local representative colors in Y or Cb or Cr. This feature is invariant to resolution.

3. **Dominant Color**[74]. This vector describes the dominant color in the corresponding color space, the percentage of the area it covers in the image, its variance and some other relevant information.

4. **Texture**. The texture of an image can be codified using from simpler [88] to more complex techniques such as in [73].

5. **Edge Histogram**. This kind of texture represents the global distribution of edges by edge histograms of several kinds of edges.

6. **Segmentation and Shape**. Different parts of an image can be identified using several segmentation techniques and, then, each segment can be represented us-

[1]A video shot is a limited section of a video, commonly created between a fade in and fade out

ing any low-level feature. Furthermore, shapes can be identified inside an image and thus different semantics found.

In Figure 7.2 examples of some visual features are shown (obtained from [81]).

In conclusion, there exist a wide variety of features and, beforehand, it is difficult to choose which to use. As shown below, several experiments have been performed in order to help to make this decision easier and more effective.

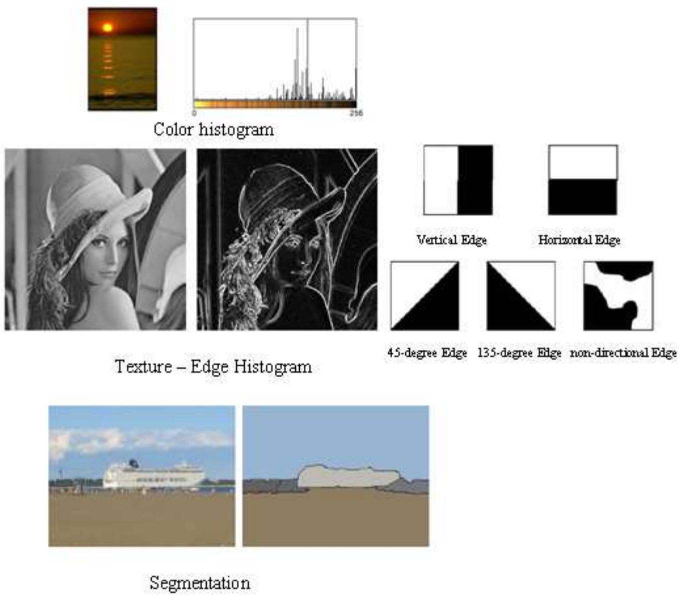

Figure 7.2: Examples of Low-Level Features.

7.4.2 Experiments on Visual Features Selection

Retrieval using low-level features faces two major problems. The first is the already mentioned *semantic gap*; and the second, the well-known "curse of dimensionality", which has been studied extensively, e.g. in [117]. To overcome this problem, solutions have been proposed in the field of multidimensional indexing structures, involving the creation of structures which allow efficient access to multimedia databases [105; 113].

In this section a piece of work is presented in which dimensionality reduction is performed for several visual features and, besides this, an exhaustive search is performed for joining visual features. The metric used for dimension reduction and feature selection is a wrapper classification to predict relevance of video shots which are represented by the description of one key-frame using the corresponding visual features. Thus, the MIR problem is projected on a classification task with a binomial class (Relevant, Non Relevant) and the objective is to find which dimensions in each visual features are the most important when predicting the relevance of a document, for several search topics.

In the following, the methodology used to solve the problem of skewed data is explained; then, starting from this solution, the problem of reducing features dimensionality is dealt with. Finally, an exhaustive search for visual feature selection and combinations is performed.

Methodology

The corpus used for evaluation is the one provided in TRECVid 2006. The TRECVID 2006 data collection consists of approx. 160 hours of television news video in English, Arabic and Chinese which were recorded in late 2005. The data set also includes the output of an automatic speech recognition system, the output of a machine translation system (Arabic and Chinese to English) and the master shot reference. The collection has 79484 shots, where each shot is considered as a separate document and is represented by low-level visual features. This experiment uses the set of 24 topics contained in the data collection, where each topic contains a judgment list of 60 to 775 relevant documents.

The video shots of the TRECVID 2006 corpus are the instances to train and classify, from which only an average of 300 shots are relevant for each search topic. So this is a highly-skewed set of shots to learn and to predict their relevance. The evaluation method used performs a 10 cross-fold validation 10 times (10x10CV). Since test sets cannot be modified and splits are made randomly in each run, the balancing of training sets needs to be performed in execution time. Since the whole database consists of about 80000 instances, 10x10CV proves to be quite a time-consuming task so the balancing method should be as light as possible, so the method used is what is referred to here as the *Alpha* method (also used in Chapter 6), which consists in randomly deleting as many non-relevant documents from the training set as indicated. If $\alpha = 100$, then the training set contains as many non-relevant documents as relevant.

The process of the experiments is as follows:

1. Experiment to choose a classifier for selection.
2. Experiment to choose the degree of *alpha* balance for training sets.
3. Experiment for Dimensionality reduction.
4. Experiment for exhaustive Visual Feature selection.

Results obtained from evaluations are assessed using three common measures in classification problems: (see Section 2.2.2) $precision$, $recall$ and $F_1 - measure$, the three of them computed for class value "Relevant".

1) First, for each visual feature[1], a 10x10CV was run over TRECVID 2006 representing each video shot with only that feature and for each of the 24 search topics in TRECVID 2006, using a balance degree of $\alpha = 100$. This cross-folder validation was performed using four different classifiers: Naïve Bayes, AODE [116], Support Vector Machines and k-Nearest Neighbor. The probabilistic classifier AODE proved to be the best trade-off between speed and performance.

2) Then, another 10x10CV was run using AODE for each visual feature and each of the 24 search topics in TRECVID 2006, using balance degrees from 0 to 100.

Results for each topic are averaged and shown in Figures 7.3, 7.4 and 7.5 for eleven different values of α and for five different visual features used to represent shots in the database.

Dominant Color and *Color Layout* values in *precision(R)* present high precisions, although that is due to outliers in one of the search topics which is related to sports (green is the most common color), so their performances cannot be expected to be that high in all topics. Thus, it is concluded that, in general, the two best performing visual features for all topics come down to be *Texture* and *Edge Histogram*.

Figures 7.3 and 7.4 reveal the common behavior of $precision$ and $recall$: as one increases the other goes down. To find a good break-even point, the $F_1 - measure$

[1]Please note that *visual feature* refers to a vector of values, where the length of such a vector depends on the kind of visual feature in question.

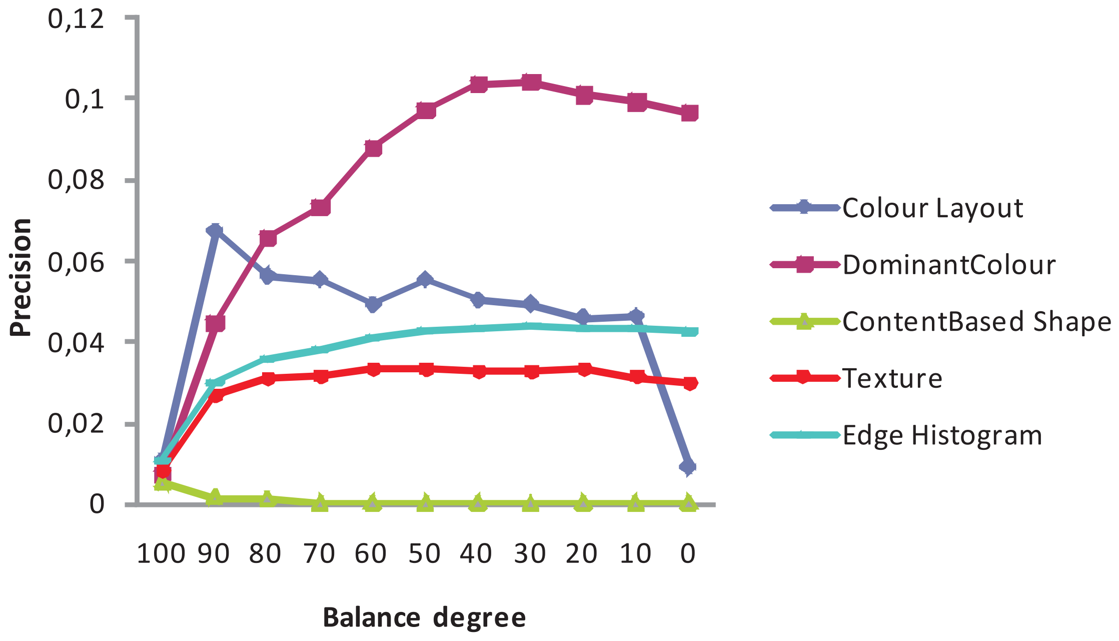

Figure 7.3: Precision for relevant shots prediction.

(Figure 7.5) is computed and the conclusion is that a balance degree of $\alpha = 50$ can be set for the dimensionality reduction experiments.

Results on Dimensionality Reduction

The quality of the used set of features is of great importance for the classifier to achieve a good performance [8]. This performance will depend of the individual relevance of each feature respect to the class, relationship among features and the existence of features which influence negatively on the classifier.

It is possible to improve the quality of the available features by performing Feature Subset Selection (see Section 2.4).

The information retrieval task is being tackled as a classification problem and, as such, it can be performed a dimensionality reduction for each visual feature. Each visual feature (Colour Layout, Dominant Colour, Content Shape, Texture and Edge Histogram) is represented by a vector of double valued features. Table 7.1 shows the number of dimensions that each visual feature contents.

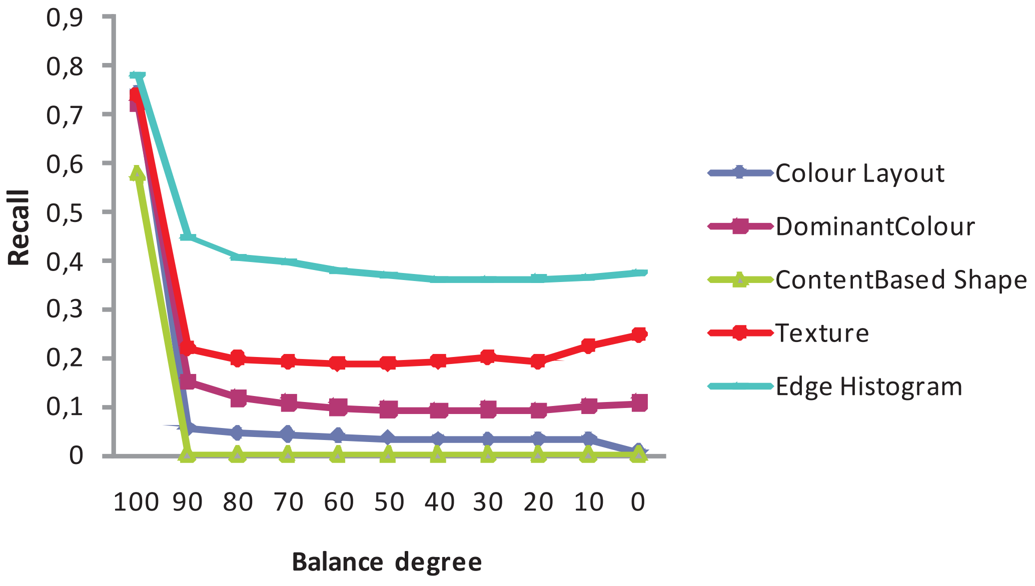

Figure 7.4: Recall for relevant shots prediction.

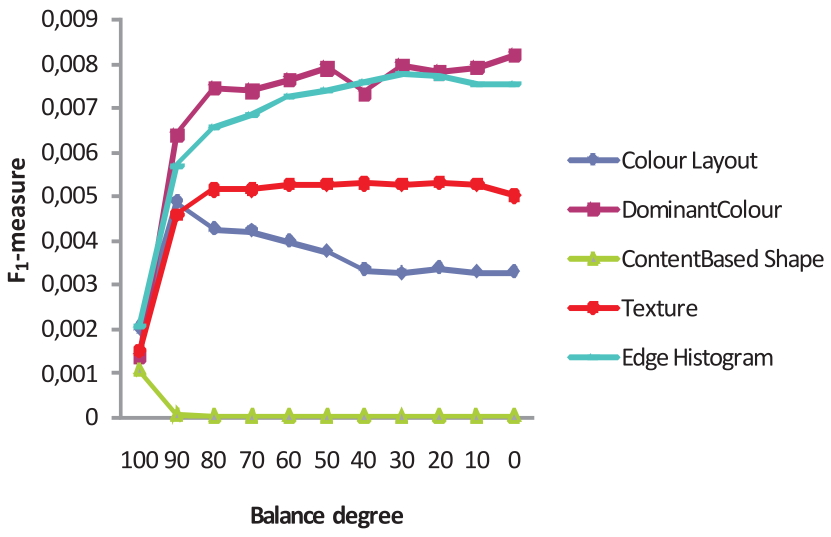

Figure 7.5: F_1-measure prediction.

Table 7.1: Length of arrays describing visual features

Visual Feature	Length
Colour Layout	10
Dominant Colour	15
Content Shape	130
Texture	62
Edge Histogram	80

A feature (dimension of a visual feature, in this case) can be regarded as an observation for a sample, and from that point of view it would be interesting to have as many observations as possible. However, a large array of observations might contain a lot of noise which leads to wrong conclusions. Besides, TRECVID 2006 is a database with a huge number of samples so no long visual features should be needed to feed the classifier. Moreover, when studying visual features instantiations to describe shots in TRECVID 2006, it is found out that some dimensions are always set to 0. So the hypothesis is that a dimensional subset selection might be helpful to improve the classifier's performance in terms of time and/or $F_1 - measure$, for the TrecVid2006 corpus. In [95] authors perform selection using a Feature Vector Reduction process on 2 Corel corpus. Although results are good, they fix the reduced vector to represent color and texture visual features without explaining why. In this study, no previous selection for any visual feature is done. Besides, no dimension nor visual feature selection has been found in the literature applied to TrecVid news corpus.

Since wrapper methods bias the results toward the wrapper classifier and the goal is to apply the results to information retrieval systems, filter metric is used to perform dimension selection. Besides, TrecVid2006 is such a huge corpus that indeed a filter metric is needed. Finally, evaluation of the goodness of selection is performed via wrapper classification. Thus, selection is a filter+metric approach.

[119] et. al present a mathematical study from where they conclude that information-based metrics as *Information Gain* are biased, favoring the selection of nominal attributes which have a higher number of states. However, a more modern study [34] performed experiments over a huge workbench and concluded that "Information Gain metric is a decent choice if one's goal is precision", which is the case since information retrieval system aim for that performance measure. Forman's work compares different information-based and statistical metrics (including chi-squre), and then concludes

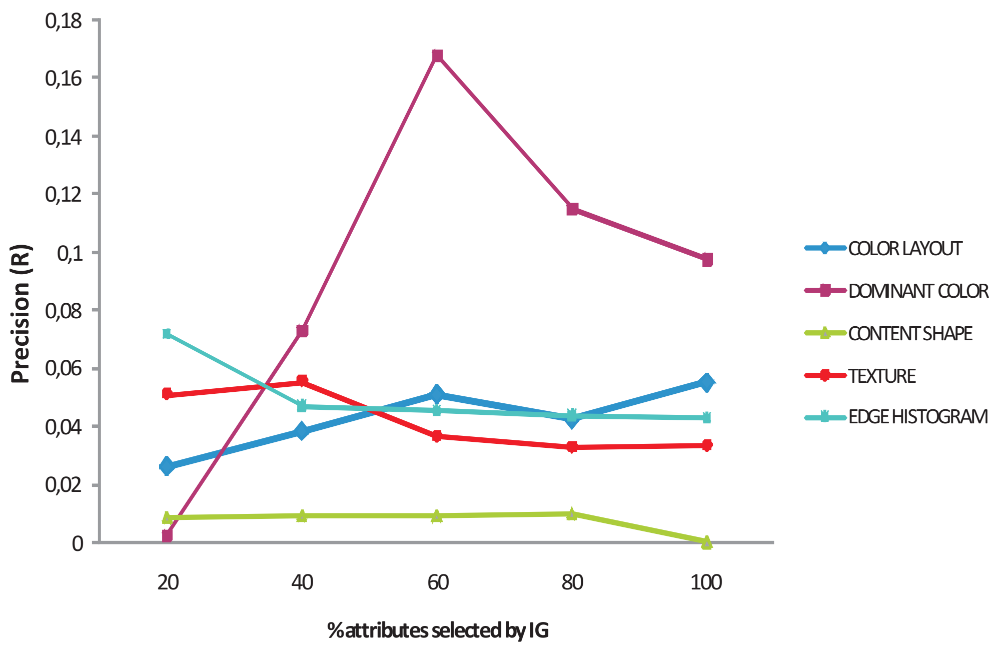

Figure 7.6: Precision (for relevant documents) over different values for P.

that“Under low skew, IG performs best and eventually reaches the performance of using all features”. Since we balance our training sets, we have a very low skew. So, based on this work, a selection by Information Gain metric respect to the class is used for feature selection.

For each visual feature, the IG value for each dimension respect to the class is used to create a ranking to know which indexes of the vector in each visual feature is more relevant respect to the class. Then, the best percentage P of features in the ranking is projected over the database and classification is performed to compute how good this new subset results to the classifier. This classification is performed as described in Section 7.4.2, and training sets are balanced setting $\alpha = 50$ as it was computed to result the best level of balance.

Several values for P are tested and again three different metrics are computed: precision(R), recall(R) and F_1-measure. For the sake of clarity it is only shown results for precision in Figure 7.6.

Results in Figure 7.6 show that the hypothesis is correct and thus a fine dimensionality subset selection can be done for visual features. Keeping just the best 40%, 50% and 60% of dimensions ordered by their IG with respect to the class makes the classifier have a slightly loss in prediction (based on precision for relevant documents) power while dividing the dimensions of visual features into halves, so information retrieval systems could be benefited by achieving a faster response to user's queries without losing quality in their final list of suggested documents.

Results on Visual Feature Selection

The hypothesis here is that the combination of two or more visual features might improve the performance of the classifier. Since there are available 5 visual features (Colour Layout, Dominant Colour, Content Shape, Texture and Edge Histogram), the search space consists of $2^5 - 1$ possible combinations. Thus, although a quite time consuming task, it is still possible to perform an exhaustive search to find out what combination of visual features makes our classifier work better.

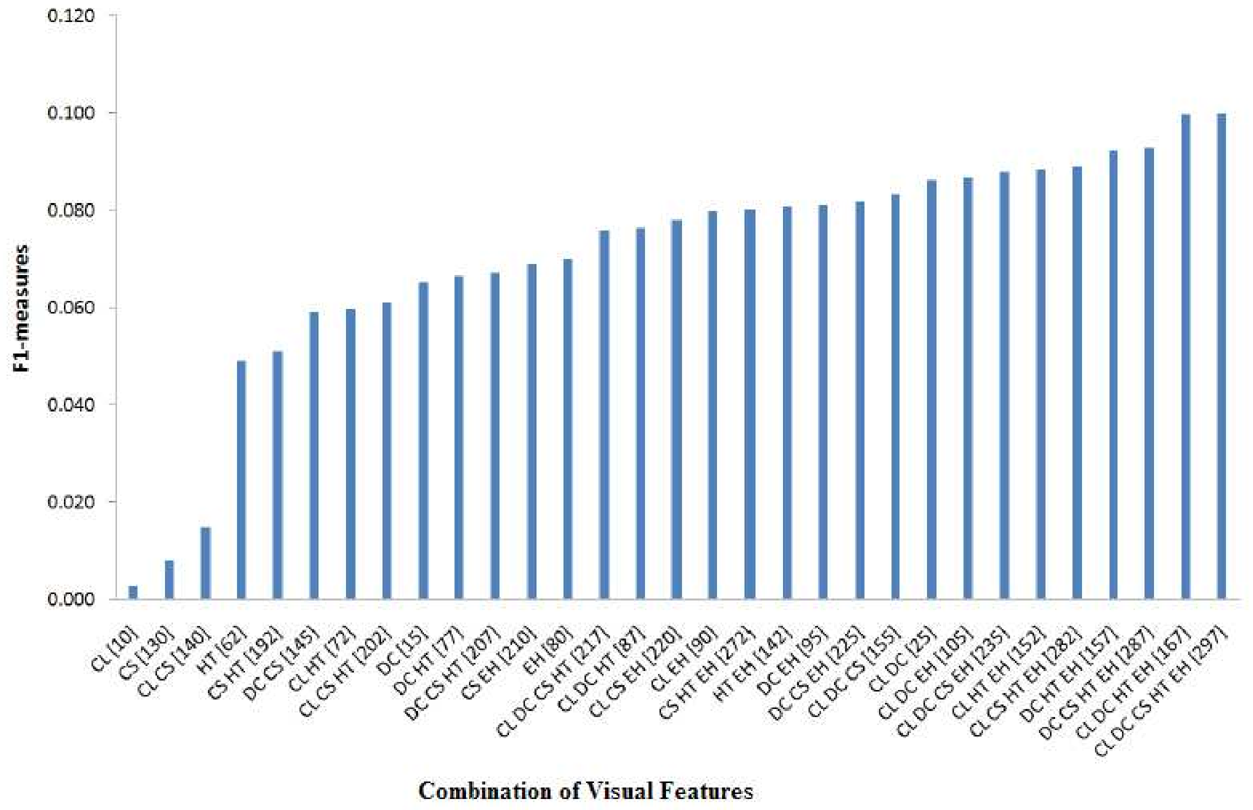

Figure 7.7: F_1-measure for all possible combinations of visual features.

Figure 7.7 shows the value for $F_1 - measure$ averaged over all the 24 TRECVID 2006 topics. It shows that the best combinations are “Colour Layout DomColour Texture EdgeHist [167]” and “Colour Layout DomColour ContShape Texture EdgeHist [297]”, both reaching $F_1 - measure = 0.1$. Results show that combination of visual features tend to improve the performance of classifier and, although this means an increase in its computational load, this score is better than any score achieved when performing feature subset selection in Section 7.4.2.

A dimensionality subset selection (the same way than in 7.4.2) for the best two combinations to check if it can kept their good performance but decreasing their high dimensionality. The selection of 40% and 60% in an IG-ranked list of the features belonging to each combination results shows that $F_1 - measure$ does not decrease for both combinations of features while their dimensionality is halved.

Conclusions

Two problems have been tackled:

1. Dimensionality subset selection. The dimensionality of visual features has been successfully reduced without decreasing the performance of the classifier, finding out that we can get rid up to the worst (based on IG) 60% dimensions inside each visual feature.

2. Visual features combination. An exhaustive search has been performed to find what combination of 5 visual features performs best (in terms of $F_1 - measure$) to predict the relevance of documents. Conclusion is that the best two combinations are all the visual features and all the visual features except Content Shape visual feature. Since these combinations derive in high dimensionality, a filter selection based on IG-ranking reduces these new dimensionalities without decreasing the $F_1 - measure$ value.

7.5 Study on Feature Construction for Video-shots Representation

Low-level visual features is one of the most widespread kind of representative features used in MIR systems. In this section, a study is presented in which four different kind of features (one of them including visual features) are used to represent video shots. As mentioned in Section 7.3, low-level features automatic extraction is most important for the speed and scalability of CBIR systems. However, several studies as in [2] search for new document representations in order to improve the performance of MIR engines. In this section a study on representation of key-frames extracted from video shots is presented and several interesting conclusions are induced. Again, the MIR problem is projected on a classification task where class is binomial ("Relevant", "Non Relevant") and each instance in the database is a key-frame of a video shot. In this study, a comparison is made on classifiers' performance using four different kind of representative features.

In order to avoid problems of overfiting and erroneous conclusions, evaluations are made over several different configurations:

- As it will be explained in more detail in Section 7.5.1,four different kind of features are used to represent shots with which the classifier is fed: features representing behavior of users during user study, features representing physical and metadata information about shots, vocabulary features and windowed vocabulary features extracted from Automatic Speech Recognition (ASR) in shots.

- The study is performed on two databases coming from two user studies, which are introduced in Section 7.5.1.

- Three different classifiers are used: probabilistic (Naive Bayes), distance-based (kNN) and vectorial-based (SVM).

- In one of these databases two different kind of relevance (two ways of deciding class value) are used: official relevance and relevance as perceived by users.

7.5.1 Methodology

In order to learn how different kinds of constructed features affect relevance prediction, data logs from two users experiments are used to construct the final datasets using different kinds of representative features, and these datasets are used for evaluation using several classifiers. This section explains how these final datasets are constructed, the different kinds of features used and how the classifiers are evaluated.

Datasets creation from user logs

It is denoted ’user study’ to refer to an experiment in which several users tested a video retrieval system searching under different topics and conditions. A log file was created from each of the users studies [115] and [114] (respectivel named “Collaborative” and “StoryBoard” user studies). Each log file contains verbose data explaining the actions each user performed (on which shots[1] actions were performed, the kind of action performed, timestamp, user condition, topic of search,...). For each query search, the user interacts with a set of shots. So, for each tuple $\langle$search, user, condition, topic, shot$\rangle$ a new instance is created from a log for the final dataset. Each instance in the final dataset consists of: tuple features ($\langle$search, user, condition, topic, shot$\rangle$), Relevance Prediction Features which are constructed to represent the shot and predict the class feature, and the class feature itself. Class features is binomial with values “Relevant” and “Non Relevant”, and refers to the relevance of the shot. For the same log, different final datasets have been constructed because different kind of features have been tested to predict relevance and additionally, different kinds of relevance decision have been tested, as explained below.

Kind of features constructed to predict relevance

As mentioned previously, an instance in the final dataset will follow the pattern

$$Tuple\ Features,\ Relevance\ Prediction\ Features,\ Class\ Feature.$$

Four different kinds of $Relevance\ Prediction\ Features$ are used: User Behavior Features, Object Features, Vocabulary Features and Windowed Vocabulary Features.

[1]In Multimedia IR systems, retrieved documents are not the whole videos but shots, where a shot is one of the splits a video can be divided into.

Table 7.2: Behavior Features used to predict shots relevance

Feature name	**Description**
ClickFreq	Number of mouse clicks on shot
ClickProb	*ClickFreq* divided by total number of clicks
ClickDev	Deviation of *ClickProb*
TimeOnShot	Time the user has been performing any action on shot
CumulativeTimeOnShots	*TimeOnShot* added to time on previous shots
TimeOnAllShots	Sum of time on all shots
CumulativeTimeOnTopic	Time spent under current topic
MeanTimePerShotForThisQuery	Mean of all values for *TimeOnShot*
DevAvgTimePerShotForThisQuery	Deviation of *MeanTimePerShotForThisQuery*
DevAvgCumulativeTimeOnShots	Deviation of *CumulativeTimeOnShots*
DevAvgCumulativeTimeOnTopic	Deviation of *CumulativeTimeOnTopic*
QueryLength	Number of words in current text query
WordsSharedWithLastQuery	Number of equal words in current query and last query

Thus, for the logs from each user study four final datasets $DS1$, $DS2$, $DS3$ and $DS4$ are derived, where the four datasets contain the same values for $Tuple\ Features$ and $Class\ Feature$ but each one contains one of the four kind of features constructed.

The User Behavior Features where designed similar to [2]. These features give information about how the user interacts with a document. In this case, the information is related to the actions the user performed through shots suggested by the information retrieval system after he/she ran a query under a concrete topic and condition. Behavior features used in this study are shown in Table 7.2 and they can be split into three groups: *Click-Through features*, which represent information about clicks the user performed on shots; *Browsing features*, which show different metrics about time spent on shots and *Query-Text features*, which count words in the current text query and make comparisons with other text queries. Note that the values for these features are computed for each tuple (⟨search,user,condition,topic,shot⟩) from the users studies logs.

Object Features are not extracted from the logs. They represent both *Low-Level Features* and *Metadata* and they are shown in Table 7.3. Using these features, the $Relevance\ Prediction\ Features$ describe the shot appearing in $Tuple\ Features$.
Metadata keeps information about length of shots and also information related to the Automatic Speech Recognition (ASR) from shots audio. Text transcripts from a shots' audio is filtered through a stop-words list and a Porter stemming filter [86], and then used to extract some statistics about the text.

Table 7.3: Object Features used to predict shots relevance

Feature name	**Description**
Color Layout	vector containing 10 values
Dominant Color	vector containing 15 values
Texture	vector containing 62 values
Edge Histogram	vector containing 80 values
Content Based Shape	vector containing 130 values
Length	Time length of shot
Words	#words in Automatic Speech Transcription from shot audio
DifferentWords	#Different words in ASR from shot audio
Entropy	Shannon entropy of ASR from shot audio

Vocabulary Features are a bag of words created from the ASR. In this case the text is not used to compute statistics about the text, but to create a vocabulary of words to perform the task of text classification. The transcripted text is also filtered through a stop-word list and a Porter stemming filter. Then, the resulting text is transformed into Weka format using a tool based on Lucene [1]. For this kind of feature the video relevance classification becomes a problem of text classification.

It is expected that video relevance classification based on ASR works relatively well due to the fact that text has more descriptive power than, for example, low level visual features. However, in the literature some complaints about using ASR can be found, as in [125] where the authors state that some speeches might not have anything in common with their respective shots, and the problem of "Coverage" (see Section 7.3).

Finally, Windowed Vocabulary Features refer to a common technique in video retrieval systems which use ASR to create the results list. This uses the same procedure performed when using Vocabulary Features but in this case the text used to construct the bag of words does not come only from the ASR of the corresponding shot but also from the n previous shots in time and the later n shots. This is call *n-Windowed ASR* and in this case it is used a 6-Windowed Vocabulary. It is expected that 6-Windowed Vocabulary features perform better than creating a bag of words from only the ASR of a single shot.

When ASR is used to create a bag of words and evaluate using a bayesian classifier, Naive Bayes is not used but the Naive Bayes Multinomial, which is recommended for text classification ([78]).

[1]http://lucene.apache.org/who.html

Table 7.4: User studies used under different combinations of kind of features and kind of relevance.

	Official Relevance	**User Relevance**
User Behavior Features	Collaborative	Collaborative & StoryBoard
Object Features	Collaborative	Collaborative & StoryBoard
Vocabulary Features	Collaborative	Collaborative & StoryBoard
W6-Vocabulary Features	Collaborative	Collaborative & StoryBoard

In the case of using Behavior Features, which are continuous values and user dependent, it is not likely to construct a dataset with repeated instances. But, when using Object or Vocabulary Features, the same shot can appear in different tuples so the $Relevance\ Prediction\ Features$ are repeated; then we would have several repeated instances in the dataset where the class feature is sometimes set as $Relevant$ and other times as $NonRelevant$. This contradiction is solved by deleting all repeated instances and setting the class feature to the most frequent value.

Kinds of relevance

Two sources of information are used to decide if a shot is relevant for a topic or not: Official Relevance and User Relevance. This means that for each final dataset, its evaluation is performed twice, once for each kind of relevance: (1) Official Relevance: Shots used in the users experiments belong to the TRECVid 2006 collection [107], which provides a list of the relevant shots for each topic based on the standard information retrieval pooling method of relevance assessment; and (2) User Relevance: In the user experiments, users could explicitly mark shots as relevant to the topic. In a dataset, a shot can be considered relevant if the user marks it as such.

One of the user studies this work is based on did not use the official TRECVid 2006 topics, so Official Relevance for that study cannot be used. Table 7.4 summarizes all the different evaluations performed for datasets obtained from each of the users studies (Collaborative and StoryBoard studies).

Relevance predictions has a different meaning depending on the kind of features and relevance used. Predicting User Relevance using User Behavior Features can be seen as predicting explicit user feedback because users marked videos (or not) after interacting with them. Predicting Official Relevance using User Behavior Features

Table 7.5: $F_1 - measure$ for datasets constructed from Collaborative users study - Official Relevance.

	Behavior	**Object**	**Vocabulary**	**W6-Vocabulary**
Nbayes/NBM	0.194	●0.06	●0.047	●0.060
SVM	0.141	●0.07	●0.052	●0.086
k-NN	0.164	●0.06	●0.046	0.172
Mean	**0.166**	**0.06**	**0.048**	**0.106**

predicts the relevance of a shot decided by a third group by actions users performed on the shots influenced by their perceptions. If Official Relevance is used when feeding our classifier with Object or Vocabulary Features values, it is assumed that low level features (as Color Layout) are meaningful enough to cross the semantic gap to high level concepts. Similarly, when predicting User Relevance using Object Features, some influence between low level features and metadata in user perception is assumed. When using Vocabulary Features, relevance prediction is similar to when Object Features are used, but in this case the semantic gap is not the problem but the “Coverage” as stated in Section 7.3, besides of becoming a text classification task.

As Table 7.4 shows, 4 datasets were created from the Collaborative user study and 2 datasets from the StoryBoard user study. For each of these datasets, evaluation was performed using each of the four kinds of feature introduced in section 7.5.1.

7.5.2 Experiments

Experiments are carried out on datasets constructed from logs obtained in two user studies. For each constructed dataset, four new datasets are derived using either User Behavior, Object, Vocabulary or Windowed-Vocabulary Features (see Section 7.5.1) to predict each shots’ relevance. For each dataset, a 5x2CV is performed (using three different classifiers ant kind of relevance (class): Official and User relevance). For each evaluation, $F_1 - measure$ is computed as a representation of classifiers’ performance. To compare the four different kind of features used to represent shots, $F_1 - measure$ is compared for each pair of representation-classifier.

Results from evaluations are shown in Tables 7.5, 7.6 and 7.7.

Mean $F_1 - measure$ for databases using Behavior Features representation always performs best. A paired Wilcoxon signed rank test [23; 121] was run with α=0.05, using, for each classifier in each table, Behavior Features as control and comparing it versus each other database for the same classifier and table. Input, for each compari-

Table 7.6: $F_1 - measure$ for datasets constructed from Collaborative users study - User Relevance.

	Behavior	**Object**	**Vocabulary**	**W6-Vocabulary**
Nbayes/NBM	0.433	●0.074	●0.094	●0.086
SVM	0.490	●0.078	●0.108	●0.095
k-NN	0.380	●0.073	●0.104	●0.145
Mean	**0.434**	**0.075**	**0.102**	**0.108**

Table 7.7: $F_1 - measure$ for datasets constructed from StoryBoard users study - User Relevance.

	Behavior	**Object**	**Vocabulary**	**W6-Vocabulary**
Nbayes/NBM	0.397	●0.313	●0.320	0.484
SVM	0.462	●0.311	●0.317	0.472
k-NN	0.601	●0.312	●0.286	●0.438
Mean	**0.487**	**0.312**	**0.307**	**0.465**

son, was the 10 values coming from the 5x2 CV evaluation. When $F_1 - measure$ for a classifier and database performed statistically less than the baseline (behavior features with the same classifier), the corresponding cell is marked with a $\bullet$ symbol. Results suggest that Behavior Features databases make classifiers usually perform statistically better than the others; this evidence supports the common idea of using collaborative retrieval to improve information retrieval systems. Another conclusion that can be extracted from result tables is that representing documents using extended vocabulary usually improves performance compared with single-document vocabulary representation.

Another interesting conclusion is that the semantic gap present in Object Features affects relevance prediction as much as the problem of coverage of manual tag labeling. Since ASR is not possible in images nor videos without voice, it can be seen as a particular case of labeling. Thus, results suggests that the aim of CBIR research community of automatic indexing of multimedia documents by their low-level features is, at least, more effective in terms of cost and effort than manually tagging.

Feature Selection

Selection has only been run on Behavior and Object Features, since selection on Vocabulary features would only return a set of words with no generalization power.

In Table 7.8 it is shown the constructed features chosen by the incremental selection. With respect to Behavior Features, we can see that features constructed to rep-

Table 7.8: Selected features when performing Incremental Wrapper-Based Selection

Behavior	Official R.	User R.	User R.
ClickFreq	x	x	x
ClickProb	x	x	
ClickDev	x	x	x
TimeOnShot			x
CumulativeTimeOnShots			x
TimeOnAllShots	x		
CumulativeTimeOnTopic			
MeanTimePerShotForThisQuery	x	x	x
DevAvgTimePerShotForThisQuery			
DevAvgCumulativeTimeOnShots			x
DevAvgCumulativeTimeOnTopic		x	x
QueryLength	x		x
WordsSharedWithLastQuery			x

Object	Official R.	User R.	User R.
Color Layout[10]			
Dominant Color[15]	1		5
Texture[62]		1	
Edge Histogram[80]	10	2	5
Content Based Shape[130]	1	1	
Length		x	
Words			
DifferentWords			
Entropy		x	

resent statistics about clicks performed are the most frequently selected. This makes sense and can be expected, since clicks can be regarded as explicit feedback about the interests of the user.

7.6 Experiments on Relevance Prediction for Video Shots

This section reports experiments for the proposed balancing methods in Section 6.3 of Chapter 6. In Section 6.4 the problem of imbalance was tackled for the task of Text Categorization, concretely that of E-mail Foldering. In this case, the databases used consist of video-shots represented using two different kind of predictive features, and the task to perform is the prediction of relevance for video-shots, given different search topics.

7.6.1 Design of Experiments

Experiments have been ran over two datasets (for these datasets construction methodology, see Section 7.5.1) which were used to train a classifier for relevance judgment. These two datasets contain the same shots which the same class but, for each dataset, a different kind of predictive features were used to represent the video-shots. The first dataset consists of Behaviour Features as suggested by Agichtein et al. [2], containing continuous values. The second dataset consists of Vocabulary Features, containing a bag-of-words where each work is an integer representing a frequency. All datasets contain a binomial class $\{$relevant, non-relevant$\}$, which is highly imbalanced towards the negative class.

As in Section 7.5.1, the aim in this section is to test different balancing methods as a preprocessing step before classification. Thus, a comparison is performed for three different approaches: *Alpha*, which consists in randomly deleting as many non-relevant documents from the training set as indicated. If $\alpha = 100$, then the training set contains as many non-relevant documents as relevant; the well-known *SMOTE* algorithm, which is a state-of-the-art balancing method in data classification; and distribution-based methods presented in Chapter 6, which perform both oversampling and undersampling and, depending of the used probability distribution gives rise to different algorithms, as explained in Section 6.3.

Both the imbalanced and balanced datasets are evaluated by performing a 5×2 CV with three different classifiers: NBayes, SVM and kNN. In the case of Vocabulary Features, NBayes is replaced with Multinomial NBayes since this is the recommended classifier in literature [78] for text documents.

The main interest is two-fold:

1. analyse if balancing helps to outperform classification using both an imbalanced dataset and a random-based balanced dataset

2. statistically assess which of the compared balancing methods is more suitable for the multimedia IR problem

Since we are dealing with skewed datasets with a binomial class, Accuracy is not an appropriate metric to measure the performance of the classifiers [118]. So the selected metrics are Precision and F_1-measure of documents tagged with the minority class (relevant documents).

Finally, a statistical comparison is ran by performing a Wilcoxon signed rank test [121] with Conf. Level $= 95\%$ to compare all methods. Two sets of twelve values are used for the test. Each set is the joined output of the three used classifiers over each of the four topics.

7.6.2 Datasets

In order to evaluate the previously introduced balancing methods, the log files of two different user studies [115] are used. In both studies, users were asked to interact with multimedia information retrieval systems and retrieve as many results as possible for four pre-defined search tasks. In both evaluations, the TRECVid 2006 video collection [107] was used where news video shots, the atomic unit of retrieval, are indexed based on the output of an automatic speech recognition system. A shot is defined as a part of the broadcast that has been created by a continuous recording from a single camera. The log files contained every key stroke and mouse click which was performed during this evaluation. They hence contain information how the participants of both studies interacted with relevant and non-relevant shots while using the system.

For every search task, a conversion was made of the log file format to two different dataset representations. The first dataset used Behaviour Features (see Table 7.2) such as a click to start playing the video shot or the playing duration. The actual relevance of the shot to the given search topic is defined as the class. Table 7.9 provides an overview of this dataset.

Table 7.9: Description of Dataset using Behaviour Features.

	#Features	**#Instances**	**Imbalance**
Topic 1	13	5011	1:19
Topic 2	13	4542	1:16
Topic 3	13	4545	1:13
Topic 4	13	4701	1:40

The second dataset uses Vocabulary Features to represent video-shots. Therefore, every shot is represented with the frequencies of terms (defined in a bag-of-words) in their audio speech, where the speech of every shots is automatically transcripted. Some shots in the corpus do not contain any speech; thus, no transcript is available for these shots. Therefore these shots are ignored, what results in fewer instances in the dataset in comparison to the behaviour feature dataset. Again, the shot's relevance was used as class. Table 7.10 provides an overview of the vocabulary feature dataset.

Table 7.10: Description of Datasets using Vocabulary Features.

	#Features	**#Instances**	**Imbalance**
Topic 1	12957	2544	1:38
Topic 2	12957	2496	1:49
Topic 3	12957	2088	1:27
Topic 4	12957	2400	1:16

The last column of these tables depicts the imbalance ratio for each dataset. A ratio of (1:*n*) means that for each relevant document, the dataset contains n non-relevant documents. It is clear that Topic 4 is the most skewed dataset for both shot representations. Therefore, it is expected that the imbalance problem affects classifiers more aggressively in this case.

7.6.3 Settings

The balancing algorithms under study in this experiments rely on various parameter settings which will be introduced in the following.

SMOTE uses the default parameters in Weka except for the percentage of minority class cardinality, which is set as the necessary percentage to get P minority class documents. Then, random under-sampling of the majority class is performed until P majority class documents remain. Distribution-based methods *Uniform, Gaussian, Poisson* and *Multinomial* need the input parameter P. If, for example, $P = 500$ then, for each training set, 500 relevant documents will be sampled from the corresponding distribution and non-relevant documents will be uniformly deleted until the training set contains 500 non-relevant documents.

The *Alpha* approach needs parameter α which indicates the percentage of non-relevant documents to be randomly removed from the training set. It was set set as $\alpha = 100$, what means to remove as many non-relevant documents as necessary to have the same number of relevant and non-relevant documents.

Since the aim of this experiment is not to oversample non-relevant documents but to create training sets with the same number of relevant and non-relevant documents,

P cannot be set to a higher value of the cardinality of non-relevant documents in any training set in the cross validation process. For example, *Topic 1* using *Behaviour Features* is a dataset with 5011 instances where about 4750 are non-relevant. Thus, since it is performed a stratified 5×2 CV, each fold will have $4750/2$ non-relevant documents, requiring P not to be greater than that value. Since it is desired to use the same P value along the four topics compared, the maximum value used in the experiments is 2000 for Behaviour Features datasets and 1000 for Vocabulary Features datasets.

7.6.4 Results and Discussion

Table 7.11 shows the mean precision P and F_1-measure for classifiers NBayes, SVM and kNN after training the imbalanced datasets using two different kind of predictive features for video shots: Behaviour and Vocabulary features. These results are used as the baseline run to compare with classification after balancing results with.

Table 7.11: Precision (P) and F_1-measure for relevant documents in imbalanced datasets.

	Behaviour		**Vocabulary**	
	P	F_1	**P**	F_1
Topic 1	.104	.142	.050	.045
Topic 2	.114	.155	.041	.029
Topic 3	.137	.166	.052	.054
Topic 4	.089	.124	.000	.000
Mean	**.111**	**.147**	**.036**	**.032**

Tables 7.12 and 7.13 list the computed precision and F_1-measure using Behaviour and Vocabulary features representation for video shots and running balancing techniques over the datasets prior to classification. The remainder of this section discusses the outcome of results using, for each kind of features, the maximum P value.

Balancing Vs. Non Balanced. In terms of precision, it is concluded that *SMOTE* and distribution-based (*Uniform*, *Gaussian*, *Poisson* and *Multinomial*) balancing methods statistically outperform the baseline using the Behaviour Features datasets. In the case of Vocabulary Features datasets both Uniform and Gaussian distributions are an exemption though. Balancing method *Alpha* does not outperform the baseline, what

Table 7.12: Performance of Balancing Methods in Behaviour Features Datasets ($P = 2000$).

	Alpha		**Uniform**		**Gaussian**		**Poisson**		**Multinomial**		**SMOTE**	
	P	F_1	**P**	F_1	**P**	F_1	**P**	F_1	**P**	F_1	**P**	F_1
Topic 1	.081	.136	.125	.076	.215	.111	.081	.046	.088	.043	.113	.168
Topic 2	.092	.159	.164	.077	.212	.162	.131	.094	.141	.098	.114	.173
Topic 3	.126	.207	.268	.078	.191	.172	.239	.113	.282	.107	.155	.229
Topic 4	.065	.116	.154	.122	.155	.142	.093	.062	.083	.063	.100	.161
Mean	**.091**	**.154**	**.178**	**.088**	**.194**	**.147**	**.136**	**.079**	**.149**	**.078**	**.121**	**.183**

Table 7.13: Performance of Balancing Methods in Vocabulary Features Datasets ($P = 1000$).

	Alpha		**Uniform**		**Gaussian**		**Poisson**		**Multinomial**		**SMOTE**	
	P	F_1	**P**	F_1	**P**	F_1	**P**	F_1	**P**	F_1	**P**	F_1
Topic 1	.028	.053	.015	.019	.076	.037	.051	.060	.053	.063	.044	.065
Topic 2	.024	.045	.018	.021	.016	.022	.055	.057	.082	.074	.044	.066
Topic 3	.043	.080	.020	.020	.043	.053	.084	.093	.098	.084	.081	.112
Topic 4	.006	.012	.000	.000	.000	.000	.005	.008	.004	.006	.001	.002
Mean	**.025**	**.047**	**.013**	**.015**	**.034**	**.028**	**.049**	**.054**	**.054**	**.059**	**.042**	**.061**

could indicate that randomly under-sampling the training set alone is not effective. In terms of F_1-measure, distribution-based methods under-perform. This is interpreted as distribution-based methods increase Precision while losing Recall, what makes sense with conclusions in Section 6.4 where Accuracy increased, and Accuracy is an average of Precision for each class (see Section 2.2.2).

Distribution-based Vs. SMOTE. Using the behaviour features dataset, the *Gaussian* method significantly outperforms *SMOTE*, while the *Multinomial* method outperforms *SMOTE* using the vocabulary features datasets. This can be explained by the fact that the *Multinomial* distribution is designed to sample new text documents from existing ones. Besides, Behaviour Features are continuous values which cannot be modeled with this distribution, while the Gaussian Distribution is a quite general model which fits well to this dataset. *SMOTE*-balanced classifiers, however, result in a better F_1-measure.

Value for P. Balancing was performed in training sets by transforming them into datasets with the same cardinality (P) for both positive and negative classes (see Table 7.14). It has been found that the larger P is, the better *SMOTE* and distribution-based

Table 7.14: Precision in distribution-based and SMOTE methods as P increases

	Gaussian		**Poisson**		**Multinomial**		**SMOTE**	
P	Behav	Vocab	Behav	Vocab	Behav	Vocab	Behav	Vocab
500	0.145	0.025	0.104	0.041	0.107	0.046	0.102	0.039
1000	0.170	0.034	0.118	0.049	0.131	0.059	0.109	0.042
2000	0.194		0.136		0.149		0.121	

methods perform in terms of Precision. Besides, non-random distribution-based methods improve most with increasing P value. As mentioned above, P is limited by the majority class cardinality so the maximum possible value for P was 2000 for Behaviour Features datasets and 1000 for Vocabulary Features datasets. For future works where ratio (1:1) is not fixed, a larger study of P influence would be of interest.

7.7 Conclusions and Future Work

When users use an information retrieval system, their implicit actions while interacting with the system can be exploited to predict relevance of documents. This feedback can be used to train a supervised classifier that effectively predicts such relevance. Considering classical information retrieval experiences, users will by far interact more with non-relevant rather than relevant documents. Thus, the main problem of using such feedback data is that they are highly imbalanced. In this Chapter, this problem was addressed by evaluating various balancing methods.

The performance of six balancing methods was evaluated: state-of-the-art *SMOTE* (directed over-sampling and random under-sampling), *Alpha* (random under-sampling) and four distribution-based methods: *Uniform*, *Gaussian*, *Poisson* and *Multinomial* (directed over-sampling with replacement and random under-sampling). Results suggest that balancing training sets using distribution-based methods result in a higher Precision in comparison to the other methods. More precisely, the *Gaussian* Distribution method provides the best balancing for continuous features (Behaviour Features) while the *Multinomial* Distribution method is best for text-based features (Vocabulary Features).

As future work, it would be interesting to search for optimum ratios of (relevant : non-relevant) documents instead of fully balance training sets to ratios (1:1).

7.8 Conclusions

One of the main problems in Multimedia Information Retrieval is the known as "semantic gap"; that is, the difficulty to link the low-level visual features describing an image with the high-level concepts of that image. However, representation of multimedia documents using low-level features is very useful in order to perform automatic indexation of images and videos, and that is a task relative to Content-based Information Retrieval. Besides, there exist a wide amount of complementary multimedia documents representations which try to improve the performance of MIR systems.

In this Chapter, visual feature selection and dimensionality reduction has been performed concluding that combination of visual features results in an important increase in performance (measure in F1-measure terms). Besides, dimensionality reduction maintains the same performance reducing up to one half of the visual features array.

Besides, a study on four kinds of features representative for key-frames (still images) on video shots has been performed, concluding that Behavior Features is a very helpful representation and, thus, results suggest that collaborative retrieval can lead to good performance of MIR systems; concretely, after performing feature selection, *Click-Through* features resulted the most frequently selected of the Behavior Features set. Respect to CBIR, both Object Features and Vocabulary Features seem to provide similar performance, what can be interpreted as the fact that the semantic gap present in Object Features affects relevance prediction as much as the problem of coverage of manual tag labeling.

Respect to balancing of training sets using distribution-based methods applied to a Multimedia Information Retrieval problem (video-shots relevance prediction), conclusions are similar to those obtained when balancing was applied to Text Categorization in Chapter 6: the proposed Distribution-based methods outperform SMOTE. Besides, other conclusion which can be extracted from both works is that when datasets are represented using continuous-values features, the Gaussian distribution is more optimal, while in the case of bag-of-words representation, the Multinomial contribution logically performs better.

Chapter 8

Influence of Context on Classifiers in MIR

8.1 Summary

This Chapter presents a relatively new field of study in IR: influence of *context* in retrieval. The semantics of "context" may vary in several layers, and along the next two sections and introduction and state of the art is presented. Then, experiments are reported with conclusions about the effect of context on classifiers's performance for relevance prediction; concretely, in this case "context" refers to the way databases are created, and the multimedia documents used for evaluation are key-frames representative of video shots.

8.2 Introduction to Context

Multimedia databases have become a reality and as such, the need has arisen for effective multimedia information retrieval systems that work as accurately and fast as possible. Much research has been carried out on this problem from different points of views: ranking algorithms, feature construction, collaborative retrieval, etc., but unfortunately the performance of Multimedia Information Retrieval (MIR) systems is still far from that of text Information Retrieval (IR) due to the semantic gap: there is a discontinuity between low level visual features and the semantics of the query.

In recent years a new point of view has arisen to better understand both text and multimedia IR process, the so-called *context*. The main idea behind this is two-sided: (1) relevance of documents might change dependently on the context on which search is being performed and (2) some contexts might make it easier for classifiers to learn an appropriate model to predict relevance.

Context refers to a wide variety of situations. For example, the relevance of a document may change if a user is web searching from a desktop computer or from a mobile device. Moreover, the relevance may be different depending on the place the user is in the moment of search; thus, if her query is “typical restaurants”, suggestions of results will be different if she is in France or in Italy. Besides, is she adds the adverb “near”, the semantics of such term is different if she is traveling by car or on foot.

Context does not refer just to the physical properties of the user’s situation. Context may also be related to user’s interests, web searching skills, economical situation,. . . Or, on the other hand, context may refer to the difficulty of the search topic, previous searches by other users for the same topic, . . . In conclusion, “context” is a very generic work for its purpose but the main idea behind this word is clear: improve performance of MIR systems by adding extra information which is not present in the representation of documents but “around the search”.

8.3 Related Work on Context

A series of forums have been held to address aspects of context in information seeking and retrieval [18; 20; 21; 44; 45; 99]. The advances reported in the forums ranged from theoretical such as creating a taxonomy of contextual features, to empirical such as deriving new context from environments, to constructive such as new applications that exploit context. Our work aims to make a methodological advance in this area by developing a framework to measure the impact of contextual factors. Therefore, this section discusses different approaches taken to measure the impact of context for modeling document relevancy.

One way to examine the impact of contextual features is to investigate the factors that influenced people’s relevance judgements. For example, [5] discussed two sets of semi-structured interviews carried out to establish the criteria used to judge document relevancy. The study identified ten criterion categories common to both the interviews. Their results highlighted that people employed non-topical factors such as quality of sources for relevance judgements. [112] observed interactive search sessions to extract the factors that influenced people's relevance judgements. They identified five groups of influential factors based on 24 participants performing three different search tasks on the Web. Their results suggest that non-textual elements in documents such as structure and visual features are affective to people's relevance assessments.

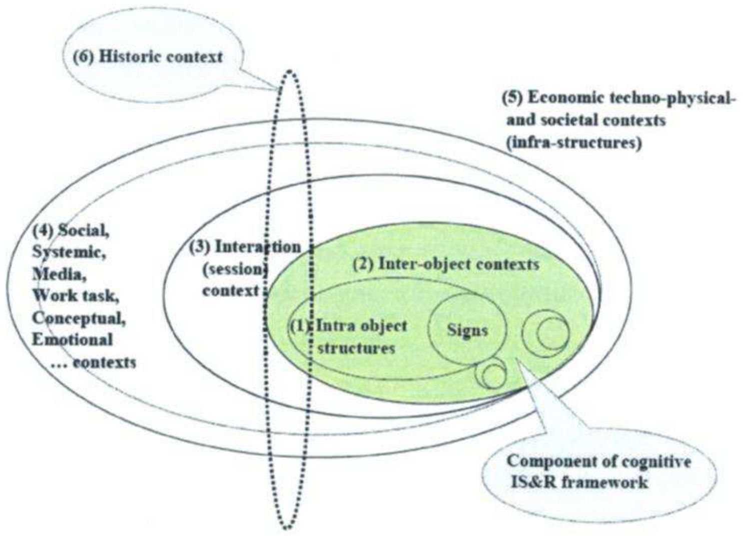
(6) Historic context
(5) Economic techno-physical- and societal contexts (infra-structures)
(4) Social, Systemic, Media, Work task, Conceptual, Emotional ... contexts
(3) Interaction (session) context
(2) Inter-object contexts
(1) Intra object structures
Signs
Component of cognitive IS&R framework

Figure 8.1: Context stratification for IR [46].

Another way to examine the impact of contextual features is to investigate their effect on searching behaviour. For example, [56] studied the effect of tasks and searchers on reading time of retrieved documents. Their experiments show a significant correlation between the contextual features and searching behaviour. Reading time was found to vary across search tasks as well as individual searchers. This suggests that reading time can be unreliable to model relevance without context. A similar approach was taken by [120] who studied the effect of topic complexity, search experience, and search stage in the performance of implicit relevance feedback. Implicit feedback was used to suggest expansion terms in the study. A mixture of measures such as subject assessments, take-up rate of suggested terms, retrieval effectiveness was used to capture the effect of the contextual features. Their study shows that all three factors affect the utility of implicit relevance feedback.

A distinct approach taken by [35] was to model document relevancy based on a history of interactions. They analysed a couple of dozens of user interactions and explicit relevance judgements to construct predictive Bayesian models. It can be seen that the accuracy of relevance prediction of the models was used as a measure of impact in their study. An advantage of their approach is the number of variables that can be investigated. While the other approaches can examine two or three factors at a time, the classifiers can allow us to investigate a large number of potentially effective factors. A disadvantage is that the dependency of features in the generated models is not always clear or interpretable.

Another way to find a dependency between contextual features is to measure the frequency of their co-occurrence in search environments. For example, [36] looked at two contextual features, document genres and work tasks, to find the dependency between them. The use of documents in a software engineering workplace was analysed in their study. The experiments show that there is a significant correspondence of document genres with the types of work tasks, suggesting that one can learn relevant genres by understanding the roles and tasks in an orgnisation.

Comparing to these existing studies, our work has the following characteristics. First, like [112], we measure the impact of context based on searchers' relevance assessments, and besides we use official relevance lists of accessed documents. This is because relevance judgements are a fundamental process in search, and also we are interested in a better relevance modeling using context. Second, we use a set of classifiers to model document relevancy. This allows us to go beyond the subjective assessments or simplistic frequency to measure the impact of contextual features on document relevancy. Finally, the proposed approach enables us to understand the dependency of contextual features, a similar objective to [36], in the same single framework.

8.4 Experiments on Context Influence on Classifiers for Relevance Prediction

The goal of the experiments shown in this chapter is to measure how context influences classifiers when trying to predict relevance of multimedia documents. Concretely, experiments are carried on video shots represented by their key-frame and the task performed is binary supervised classification, where class to predict is "Relevant" or "Non Relevant" for a given search topic.

In this case, "context" refers to how a database is created; for example, a database created with shots accessed by experimented users or a database created when searching easy topics. As it will be explained below, the final objective is to identify “cohesive contexts”.

8.4.1 Methodology

As explained in Section 7.5.1, experiments use data logs from two users experiments and, from these logs, the final datasets are constructed, one for each of four kinds of features and each of two kinds of relevance. The reader is referred to that section for further explanation on the process.

Then, each of the datasets constructed is split in several parts according to a given context and classifiers are evaluated and compared to the baseline (dataset with no context split).

Contexts in Databases Used

A deeper explanation on each of the two users studies used in the experiments is needed in order to understand the identified context.

- *Collaborative study*. In this study, users where grouped into pairs and searched for shots relevant to four TrecVid 2006 [107] topics under four different conditions: user A could see what user B was doing, user B could see user A, both users could see each other and, lastly, both users performed a search independently. Thus, conditions can be: Watching, Watched, Mutual, Independent. So $Tuple\ Features$ constructed from the log of this study are: User, Topic, Condition, and Shot. $Tuple\ Feature$ are used to make the context splits, but at the time of performing evaluation on any dataset they are removed in order to predict relevance just based upon the constructed $Relevance\ Prediction\ Features$.

- *StoryBoard study*. In this study, users had to use two different interfaces (a common interface as baseline and a storyboard-style interface), to search for shots relevant to two different non-TRECVid topics. $Tuple\ Features$ created from these logs are: User, Topic, System, Run and Shot. Where System is either the BaseLine interface of the designed StoryBoard interface, and Run shows which system was used the first time and which second.

Splitting into contexts

Each instance in datasets is made of

$$Tuple\ Features,\ Relevance\ Prediction\ Features,\ Class\ Feature.$$

where tuple features describe some context information as: userID, search topic, search condition,... All these are nominal features and, by selecting one of these tuple features the dataset can be split into as many possible values as the selected nominal attribute might take. This is what is called in this work as *attribute-based context* split.

Besides, another context split method is possible which is referred in this work as *property-based context*. For example, *experience* context can be made of two subgroups: less experienced users in the user study and more experienced users. This kind of information needs to be obtained via a questionnaire completed by the users. Thus, if a user study is performed with 4 people and we learn from their answers in the questionnaire that users {id1,id3} are the least experienced meanwhile users {id2,id4} come down to be the most experienced, the dataset can be split based on those two subgroups, using the *userID* tuple feature.

Finally, a third kind of context split is performed which is referred to as *mixed context*, being the result of splitting by two desired context (whereas *attribute-based* or *property-based*), being the number of splits the cartesian product of the possible splits for each of the chosen contexts. In Tables 8.1 and 8.2 contexts constructed for each user study are shown and described.

Evaluation of Classifiers Under Contexts

Evaluation to compute a baseline is performed without using the $Tuple\ Features$ so that the evaluation is totally free of context differentiation (or all contexts together, we may say), using each of the four kind of constructed features. Tables 8.1 and 8.2 show all the different contexts defined for Collaborative and StoryBoard users study, respectively; for example, in Table 8.1 we see that the dataset can be split based on context *User Experience*. This split would result in 2 datasets, one with shots accessed through the users study by less experienced users and another dataset containing shots accessed by more experienced users.

Thus, for each context shown in these tables, each dataset is split into as possible splitting values the corresponding context contains. Thus, results obtained for each

Table 8.1: Contexts costructed for Collaborative user study

Context(#splits)	**Type**	**Split name**	**Description**
Condition (4)	Attribute-based	Individual Watched Watching Mutual	User searches on his own. User is watched while searching. User watches sb. else also searching. Two users watch the other's search.
User Experience (2)	Property-based	Less Exp. More Exp.	Less searching experience. More searching experience.
Task difficulty (2) (as perceived by user)	Property-based	Easy Difficult	Easy for users. Difficult for users.
Condition& User Experience (8)	Mixed	Individual&less Individual&more Watched&less Watched&more Watching&less Watching&more Mutual&less Mutual&more	Mix of corresponding descriptions
Condition& Task Difficulty (8)	Mixed	Individual&easy Individual&diff. Watched&easy Watched&diff. Watching&easy Watching&diff. Mutual&easy Mutual&diff.	Mix of corresponding descriptions
User Experience& Task Difficulty (4)	Mixed	less&easy less&diff. more&easy more& diff.	Mix of corresponding descriptions

context will be compared against the previously computed baseline to find how context affect classifiers.

Accuracy is not used for evaluation because, although it is a standard metric used to evaluate the predictive power of classifiers, the tests sets are so unbalanced that computing Accuracy is roughly the same as computing recall for non relevant documents. Although training sets are balanced (using the method referred in Section 7.4.2), test

Table 8.2: Contexts constructed for storyBoard user study

Context(#splits)	**Type**	**Split name**	**Description**
User Experience (2)	Property-based	Less More	Less searching experience. More searching experience.
System (2)	Attribute-based	BaseLine StoryBoard	Ordinary search system. Improved search system.
Search Task (2)	Attribute-based	A B	Difficult search task. Easy search task.
Run (2)	Attribute-based	Run1 Run2	BL system was used first BL system was used first second
System& Task (4)	Mixed	BaseLine&A BaseLine&B StoryBoard&A StoryBoard&B	Mix of corresponding descriptions
System& User Experience (4)	Mixed	BaseLine&less BaseLine&more StoryBoard&less StoryBoard&more	Mix of corresponding descriptions
Task& User Experience (4)	Mixed	A&less A&more B&less B&more	Mix of corresponding descriptions

sets are not: if a classifier always marks documents as belonging to the majority class value, accuracy would be incredibly high but documents belonging to the minority class values would never be correctly predicted. For information retrieval systems, documents belonging to minority class value (relevant documents) are what is needed to predict correctly so accuracy on its own is not an appropriate metric.

8.5 Experiments

This section shows the results obtained when performing classification with three different classifiers. Evaluation is performed over two databases created from 2 users studies. For each database, evaluation is performed using one of three kinds of features (see Section 7.5.1), with the Collaborative users study represented twice, once for each kind of relevance. Windowed Vocabulary features have not been used because we they add an extra context factor and it is desired to avoid any contextual bias in the

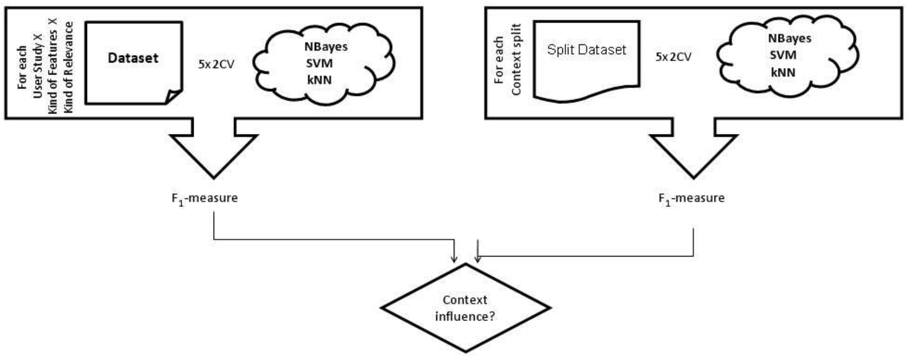

Figure 8.2: Diagram showing the process of datasets creations and evaluation to assess how context affects classifiers.

results: the additional vocabulary of nearby shots works as an extra help for classifiers, which is not desirable because the aim is to compute context influence without any extra factors.

Results are shown in two tables. Table 8.3 shows results obtained working on Collaborative user study log, and Table 8.4 shows results for StoryBoard user study log.

First row in both tables shows mean $F_1 - measure$ over NBayes, SVM and kNN, for each kind of feature, using a dataset without context split (e.g.: no context differentiation, all contexts together). Remaining rows show results for each dataset split based on the corresponding context.

Independent t-test with α=0.05 is performed to compare baseline with each context result. Thus, input in each comparison is made of two vectors containing each one a total of 30 samples (enough samples to assume normal distribution by the Law of Great Numbers). Where 10 samples come from the results obtained along the performed 5x2 CV using one of the three classifiers. Thus, 30 samples are joint (averages are being compared) and contrasted versus other corresponding 30 samples. When a cell is shadowed in Tables 8.3 and 8.4, it means that the dataset split by context has derived in a $F_1 - measure$ statistically better than the baseline it is compared to.

Results, as discussed in next section, are very meaningful and they show that contexts clearly affect classifiers' performance.

Table 8.3: Mean $F_1 - measure$ over NBayes, SVM and kNN evaluations. Collaborative user study.

	Official Relevance			User Relevance		
	Behav.	**Object**	**Vocab.**	**Behav.**	**Object**	**Vocab.**
No differentiation	**0.166**	**0.065**	**0.048**	**0.434**	**0.075**	**0.102**
Individual	0.161	0.054	0.042	0.503	0.088	0.143
Watched	0.126	0.054	0.058	0.474	0.116	0.181
Watching	0.179	0.098	0.069	0.373	0.118	0.142
Mutual	0.186	0.086	0.069	0.407	0.107	0.114
Less Exp.	0.159	0.067	0.047	0.154	0.051	0.036
More Exp.	0.172	0.078	0.075	0.616	0.144	0.182
Easy	0.172	0.105	0.069	0.431	0.109	0.137
Difficult	0.142	0.029	0.029	0.453	0.058	0.106
Individual &Less Exp.	0.141	0.062	0.046	0.132	0.035	0.025
Individual &More Exp.	0.168	0.054	0.057	0.703	0.169	0.238
Watched &Less Exp.	0.141	0.049	0.054	0.121	0.032	0.021
Watched &More Exp.	0.109	0.074	0.086	0.598	0.244	0.339
Watching & Less Exp.	0.147	0.125	0.059	0.178	0.112	0.061
Watching &More Exp.	0.214	0.099	0.082	0.538	0.175	0.242
Mutual &Less Exp.	0.168	0.093	0.066	0.173	0.077	0.068
Mutual &More Exp.	0.177	0.126	0.106	0.539	0.196	0.215
Individual &Easy	0.140	0.096	0.067	0.167	0.065	0.064
Individual &Diff.	0.166	0.024	0.032	0.705	0.131	0.198
Watched &Easy	0.138	0.070	0.060	0.625	0.183	0.296
Watched &Diff.	0.105	0.040	0.052	0.112	0.029	0.043
Watching & Easy	0.198	0.167	0.100	0.512	0.248	0.280
Watching &Diff.	0.153	0.054	0.048	0.122	0.035	0.035
Mutual &Easy	0.229	0.122	0.090	0.215	0.089	0.076
Mutual &Diff.	0.122	0.042	0.045	0.647	0.128	0.155
LessExp. &Easy	0.182	0.105	0.066	0.205	0.080	0.054
LessExp. &Diff.	0.105	0.028	0.030	0.082	0.023	0.029
MoreExp. &Easy	0.168	0.123	0.100	0.570	0.190	0.255
MoreExp. &Diff.	0.160	0.046	0.055	0.678	0.121	0.156

8.5.1 Discussion on Context Results

This section discusses the results obtained after splitting and evaluating the presented datasets under different contexts.

Table 8.4: Mean $F_1 - measure$ over NBayes, SVM and kNN evaluations. StoryBoard user study.

No differentiation	**0.487**	**0.312**	**0.307**
Less Exp.	0.437	0.286	0.309
More Exp.	0.519	0.363	0.327
BaseLine	0.522	0.377	0.356
StoryBoard	0.454	0.301	0.294
TaskA	0.415	0.282	0.254
TaskB	0.548	0.382	0.377
Run1	0.480	0.330	0.292
Run2	0.486	0.373	0.386
BaseLine & TaskA	0.375	0.215	0.237
BaseLine & TaskB	0.620	0.544	0.491
StoryBoard & TaskA	0.469	0.356	0.323
StoryBoard & TaskB	0.509	0.247	0.266
BaseLine & Less Exp.	0.499	0.355	0.387
BaseLine & More Exp.	0.562	0.440	0.362
StoryBoard & Less Exp.	0.382	0.239	0.200
StoryBoard & More Exp.	0.531	0.375	0.339
TaskA & Less Exp.	0.369	0.285	0.198
TaskA & More Exp.	0.464	0.307	0.269
TaskB & Less Exp.	0.499	0.330	0.343
TaskB & More Exp.	0.574	0.467	0.412

Is it a statistical improvement what we expect for every context split?

The answer is No. On one hand, it is expected from some contexts to contain a set of less coherent instances, like in “Difficult Tasks”, so this would lead to a poor learning for the classifier; we refer to this kind of contexts as “disturbing” contexts. On the other hand, cohesive instances are expected in some other context splits, like in “Experienced Users”. This latter kind of context can be regarded as “cohesive” context.

Thus, some contexts make classifier’s task easier and some others not or even the contrary. The aim of this experiment is to find what contexts are more cohesive and thus lead to better performance in classifiers.

When a split dataset performs statistically better, is it due to context influence or to its smaller size?

When randomly reducing the number of instances from a dataset, we are keeping the same number of properties (attributes) but having a smaller number of records for the model to learn. This makes the classifier to be trained less efficiently and thus evaluation would commonly decrease. However, if database is split based on some relevant attribute, entropy of database is decreased and a better model is expected to be learned. Thus, splitting a database based on the correct property will increase classifier's performance due to not the decrease in size but decrease in uncertainty; that is, a context split will perform statistically better when it is a cohesive context.

8.5.2 Context influence

Along this section a discussion is developed to try to understand the presented results in Tables 8.3 and 8.4. The most remarkable results obtained from the experiments are pointed out:

- User Experience. Results show, explained in next itemization, that "More Experienced" is a cohesive context which helps the classifier to learn a better model for classification.
    - Collaborative User Relevance. When using Behavior features, the actions of more experienced users are consistent in their search for what they regard as relevant shots - it is easier for the classifiers to compute appropriate statistics in order to predict relevance. For less experienced users, user actions are less consistent, the classifier learns a worse model, and the performance of the classifiers is worse. As for Object/Vocabulary features, shots selected by more experienced users are more cohesive than with less experienced users, so the statistics are learned from similar instances thus producing a greater performance.
    - Collaborative Official Relevance. Only Object features perform statistically different from the baseline, although it is worth to mention that $F_1 - measure$ for the other two kind of features is still greater than the baseline.
    - StoryBoard. In this case, it is again found a statistically improvement just for "More Experience" context. Although this is just for one kind of features, again the other two features got a low p-value and are still greater than baseline.

- Task Difficulty. In this context, it is quite clear the positive impact of splits based on Easy tasks. In most cases for “Easy” splits in Collaborative and all in “TaskB” in StoryBoard (which is the easy task), results are statistically better than classifier’s performance without context differentiation. This clearly suggests that shots accessed while searching easy tasks make more cohesive sets to predict relevance.

- Condition. Interesting results were found for this kind of context.

  – Collaborative. Results seem to suggest that search for shots that will be marked as “Relevant” by users is more cohesive when the user who will bookmark the shot is searching alone. On the contrary, when relevance is officially tagged, performance is better for contexts made of shots accessed by users helping each other or watching somebody’s actions.

  – StoryBoard. Condition can be differentiated in two cases: when users perform search using a state-of-the-art interface (BaseLine interface) or a new improved interface they face for the first time (StoryBoard interface). Results suggest that the use of an interface people are used to derive in a more coherent and compact search than search performed using for the first time a new interface, even though this is supposed to be improved.

- Run (storyBoard). This is quite a random and non deterministic context since this is just a random order in which users used the two possible interfaces. As a consequence of this, no consistent conclusion can be found nor suggested.

- Mixed Contexts. Mixed contexts are more difficult to interpret, but if interest is focused on searching for corroboration of conclusions stated above, the same conclusions can be found again in mixed contexts. Thus, in Collaborative user study, statistical improvements are again achieved mostly when one of the mixed split contexts is “More Experienced” of “Easy”. Besides, in StoryBoard user study, statistical improvement is achieved for all kind of features when we find together mix of two cohesive contexts: “BaseLine&TaskB”, “BaseLine&More Experienced” and “TaskB&More Experienced”.

- It is also quite remarkable that no statistical improvement has been achieved for any kind of feature for context splits: “Less Experienced”, “Difficult”, “Task

A”(difficult task’) nor “StoryBoard” (first contact with a new interface). So this suggests that they are disturbing contexts which decrease classifiers’ performance.

8.6 Conclusions

Besides multimedia documents representation, new fields of study have arisen in order to improve performance of MIR systems. One of them is the study of “context influence”. In this Chapter, an introduction has been made to literature on context and a deep study has been presented on context influence on classifiers for document relevance prediction.

These experiments have searched for cohesive contexts which positively influence on classifiers. In order to not get biased conclusions, experiments were run on two user studies, using two kind of relevance and three different kind of features to represent shots, plus three different classifiers. Results support the hypothesis that classifiers learn a better model when datasets on which they are evaluated have been constructed using shots accessed in cohesive contexts. Several cohesive contexts have been found; thus, shots accessed by more experienced users, or during an easy task, or using a search interface users are used to, makes cohesive sets of shots to learn a better model.

Part IV

Conclusions and Future Work

Chapter 9

Conclusions and Future Work

9.1 Conclusions and Future Work

Apart from Part I, which presents an introduction to supervised classification and feature selection, in this thesis several methodological contributions haven been presented for supervised pre-processing of high-dimensional text and multimedia databases: methods for feature selection and methods for instance selection and re-sampling.

Part II presents several proposals with experiments run on high-dimensional text databases. Chapter 3 studies new criteria for the Incremental Wrapper Subset Selection (IWSS) for feature subset selection methods to decide the inclusion of a new attribute in the subset of selected subsets. Furthermore, a proposal is made to add the possibility of not just adding a new feature but replacing it by some of the features in the selected subset; this option provides more compact subsets without decreasing accuracy. Moreover, an early stopping criterion is suggested which effectively stops the search without decreasing the performance of the incremental algorithm. Finally, the Naïve Bayes classifier is embedded inside the IWSS algorithm such that performance stays the same but the wrapper nature of the incremental search is maintained while the complexity is reduced to just that of a filter nature. Thus, execution time is drastically reduced while keeping the advantages of incremental wrapper search, and final selected subsets are more compact.
While Chapter 3 improves IWSS in its wrapper search, Chapters 4 and 5 present two proposals for extending the search space of the IWSS algorithm by altering the filter

ranking over which the wrapper search is run. Chapter 4 proposes the embedding of a GRASP search by utilizing a stochastic filter ranking instead of a deterministic one, and several methods are studied for choosing the best solution found by GRASP. On the other hand, Chapter 5 proposes a technique for detecting features at later positions in the ranking which become relevant after some features from earlier positions have been selected; then, these low-ranked features are re-ranked to earlier positions and so the wrapper search can stop sooner. This results in more compact subsets as well as a drastic reduction in number of evaluations needed; and this conclusion is corroborated by a vast number of experiments and statistical comparisons.
Chapter 6 focuses on improving the e-mail foldering performance of the Naïve Bayes Multinomial (NBM) by studying re-sampling methods of training sets based on different distributions. Experiments prove that our suggested methods statistically improve the performance of NBM.

In Part III, Chapter 7 presents a study of different kinds of features to represent multimedia documents for relevance prediction under different search topics, with the result that features which represent the actions of users while web searching are very relevant for implicit relevance feedback and, thus, they can be useful for collaborative search engines. In addition, it tests balancing methods suggested in Chapter 6 using the different kinds of representations. Finally, Chapter 8 presents a study of context influence on user while performing the search. The results give evidence to conclude that documents accessed during a search by expert users, or under easy search topics, construct cohesive sets of documents which benefit the learning stage of classifiers. And, thus, this gives a new criterion for selection of instances: to select those instances which are representative of documents coming from cohesive contexts.

As future work, it would be interesting to extend the embedding of probabilistic classifiers in the IWSS algorithm besides that of Naïve Bayes. Furthermore, since embedding a GRASP search in IWSS has proved to be so profitable (more compact subsets and lower number of evaluations), new GRASP proposals could be studied.
Finally, Chapter 8 gives a criterion for selecting instances which come from real-world searches, but this is not applicable to synthetic databases; so it would be interesting to investigate other ways of detecting cohesive sets of instances besides the difficulty of search or the experience of users.

9.2 Contributions to Literature

Publications:
I have actively collaborated in the following works:

[1] Bermejo, P., Gamez J.A. and Puerta J.M. (2010). Improving Incremental Wrapper-based Feature Subset Selection by Using Re-ranking. In IEA/AIE 2010. Proceedings *23rd International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems*, Cordoba, Spain. To appear.

[2] Bermejo, P., Hopfgartner, Gamez J.A., Puerta J.M. and Jose J. (2009). Comparison of balancing techniques for multimedia IR over imbalanced datasets. In ISCIS 09. *Proceedings of the 24th International Symposium on Computer and Information Sciences*, Morphou, Cyprus.

[3] Urruty T., Bermejo P. and Jose. J. (2009). Feature selection to improve indexing of multimedia data. In ISPS09 - *Proceedings of the 9th International Symposium on Programming and Systems*, Argel.

[4] Urruty T. and Bermejo P. (2009). Simulated evaluation of faceted browsing based on feature selection. *Multimedia Tools and Applications: Special Issue on Emerging Multimedia* (MTAP).

[5] Bermejo, P., Gamez, J. A. and Puerta, J. M. (2009). Incremental wrapper-based subset selection with replacement: An advantageous alternative to sequential forward selection. In *Proceedings of 2009 Symposium on Computational Intelligence and Data Mining* (CIDM), pages 367 to 374.

[6] Bermejo, P., Joho, H., Jose, J.M. and Villa, R. (2009). Comparison of feature construction methods for video relevance prediction. In *Proceeding of the 15th International Multimedia Modeling Conference* (MMM09), pages 185 to 196.

[7] Urruty T. and Bermejo P.. (2008). Feature subset selection for efficient indexing. In *3rd International Conference on Semantic and Digital Media Technologies* (SAMT2008).

[8] Bermejo, P., Hopfgartner, F., and Jose, J. M. (2008). Automatically creating and comparing features and contexts. In *Second International Workshop on Adaptive Information Retrieval* (AIR08), pages 27 to 31.

[9] Bermejo, P., Gamez, J., and Puerta, J. (2008). On incremental wrapper based attribute selection: experimental analysis of the relevance criteria. In IPMU08: *Pro-*

ceedings of the 12th Intl. Conf. on Information Processing and Management of Uncertainty in Knowledge Based Systems., pages 638 to 645.

[10] Bermejo, P., Gamez, J., Puerta, J., and Uribe, R. (2008). Improving knn-based e-mail classification into folders generating class balanced datasets. In IPMU08: *Proceedings of the 12th Intl. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 529 to 536.

Following works are under journals review:

[11] Bermejo. P, Joho H., Jose J. and Villa R. Comparison of video documents representation and study of context influence on classifiers. *Information Processing & Management. under 2nd review.*

[12] Bermejo, P., Gamez, J. A., and Puerta, J. M. Improving incremental wrapper-based subset selection via replacement and time optimization. Under review in *Pattern Recognition*.

[13] Bermejo, P., Gamez, J. A., and Puerta, J. M. Improving the performance of naive bayes multinomial in e-mail foldering by introducing distribution-based balance of datasets. Under review in *Expert Systems with Applications*.

Other works not directly related to this thesis:

[14] Vivo A., Bermejo P. and Tarrata P. (2009). Predicción de cancer de prostata mediante clasificacion de microarrays y cuestionario clinico (best communication award). *XIV Congreso Castellano-Manchego de Médicos Generales y de Familia*.

[15] Bermejo, P., Gamez, J. A., and Puerta, J. M. (2007). Attribute construction for e-mail foldering by using wrappered forward greedy search. In *9th International Conference on Enterprise Information Systems*, pages 247 to 252.

[16] Bermejo, P., Gamez, J. A., and Puerta, J. M. (2007). Construcción de atributos: un caso de estudio en clasificación de correo-e. In *V Congreso Español de Metaheuristicas, Algoritmos Evolutivos y Bioinspirados*, pages 555 to 562.

[17] Lopez V., Olivares O., Orozco L., Bermejo P. and Pedro P. (2005). E*nvironmental wireless sensors networks. In I Spanish Congress in Computer Science*.

Appendix A

Re-ranking Criteria

DataSet	SFS		B=5		B=10		B=20		B=30		B=40		B=50	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	83.87	5.9	82.26	2.2	85.48	2.2	83.87	2.4	83.87	2.4	83.87	2.3	83.87	2.3
Leukemia	87.50	3.2	87.50	1.9	87.50	1.9	87.50	1.9	87.50	1.9	87.50	1.9	87.50	2.0
Lymphoma	83.33	7.1	72.92	4.7	75.00	5.6	80.21	5.6	81.25	5.7	78.13	5.9	76.04	5.9
DLBCL	80.85	3.6	87.23	1.5	82.98	1.6	80.85	1.7	80.85	1.7	80.85	1.7	80.85	1.7
Prostate	75.00	5.4	73.53	3.2	75.74	3.4	77.21	4.2	80.88	4.7	80.15	4.7	83.09	4.8
Lung	93.92	2.5	96.69	2.2	96.69	2.2	97.24	2.4	97.24	2.4	97.24	2.4	97.24	2.4
GCM	58.42	18.3	51.58	7.4	53.68	10.3	57.89	10.9	57.89	11.8	60.00	13.4	62.11	14.2
Arcene	68.00	4.6	71.00	2.6	70.00	3.7	72.00	3.8	73.00	4.3	71.00	4.3	69.00	4.3
Madelon	60.75	6.5	61.65	2.0	60.75	3.4	61.25	4.8	60.25	5.9	60.10	5.6	60.50	6.2
Dorothea	91.25	13.2	94.25	3.0	93.25	4.0	93.38	5.0	93.00	5.3	92.88	5.3	92.88	5.3
Dexter	76.00	13.8	82.67	8.5	81.00	10.1	83.00	9.8	82.67	9.7	82.67	9.8	82.67	9.4
Gisette	94.05	26.9	86.20	2.7	88.47	6.4	90.77	10.8	91.62	15.9	92.25	16.5	92.57	17.1
Geom. Mean	*78.51*	*7.1*	*77.82*	*3.0*	*78.16*	*3.8*	*79.54*	*4.4*	*79.93*	*4.7*	*79.68*	*4.8*	*79.85*	*4.9*
Arith. Mean	*79.41*	*9.3*	*78.96*	*3.5*	*79.21*	*4.6*	*80.43*	*5.3*	*80.84*	*6.0*	*80.55*	*6.2*	*80.69*	*6.3*

Table A.1: Results using Naive Bayes classifier, SFS selection algorithm and CMIM-based re-ranking with block sizes B.

DataSet	**IWSS^2**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	80.6	3.8	80.6	2.8	83.9	3.0	82.3	3.2	82.3	3.3	82.3	3.3	82.3	3.3
Leukemia	87.5	2.5	87.5	2.0	87.5	2.4	87.5	2.4	87.5	2.4	87.5	2.4	87.5	2.5
Lymphoma	76.0	8.8	66.7	6.3	75.0	7.6	76.0	7.9	77.1	8.0	75.0	8.1	77.1	8.2
DLBCL	85.1	1.9	89.4	1.5	87.2	1.6	85.1	1.7	85.1	1.7	85.1	1.7	85.1	1.7
Prostate	77.9	11.1	72.1	4.1	74.3	4.0	77.9	5.6	74.3	7.3	72.1	7.8	74.3	8.0
Lung	97.2	2.7	96.7	2.2	96.7	2.4	97.2	2.7	97.2	2.7	97.2	2.7	97.2	2.7
GCM	64.2	36.6	54.2	12.3	60.0	19.8	62.1	21.4	65.3	22.5	64.2	24.4	64.7	27.4
Arcene	70.0	13.4	70.0	3.5	68.0	5.1	70.0	6.8	70.0	7.0	70.0	7.8	69.0	7.8
Madelon	59.9	13.3	61.3	2.7	60.9	4.8	60.3	7.1	59.8	8.0	60.0	10.1	59.6	11.4
Dorothea	93.5	7.4	93.9	2.8	94.1	3.6	94.4	3.8	94.0	4.3	93.9	4.3	93.8	4.5
Dexter	81.0	19.6	81.7	11.9	83.7	13.1	83.7	14.8	81.3	15.7	83.0	15.2	80.7	14.9
Gisette	94.7	112.6	88.7	18.3	92.3	41.3	93.7	62.6	93.9	69.5	94.4	82.0	94.1	77.2
Geom. Mean	*79.81*	*9.45*	*77.41*	*4.24*	*79.34*	*5.52*	*79.97*	*6.50*	*79.81*	*6.94*	*79.51*	*7.32*	*79.59*	*7.49*
Arith. Mean	*80.6*	*19.5*	*78.5*	*5.9*	*80.3*	*9.1*	*80.9*	*11.7*	*80.6*	*12.7*	*80.4*	*14.2*	*80.4*	*14.1*

Table A.2: Results using Naive Bayes classifier, IWSS^2 selection algorithm and CMIM-based re-ranking with block sizes B.

DataSet	**IWSS$_r^2$**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	83.9	2.8	82.3	2.2	85.5	2.2	83.9	2.4	83.9	2.4	83.9	2.3	83.9	2.3
Leukemia	87.5	2.0	87.5	1.9	87.5	1.9	87.5	1.9	87.5	1.9	87.5	1.9	87.5	2.0
Lymphoma	80.2	5.9	72.9	4.7	75.0	5.6	80.2	5.6	81.3	5.7	78.1	5.9	76.0	5.9
DLBCL	80.9	1.8	87.2	1.5	83.0	1.6	80.9	1.7	80.9	1.7	80.9	1.7	80.9	1.7
Prostate	78.7	7.0	73.5	3.2	75.7	3.4	77.2	4.2	80.9	4.7	80.1	4.7	83.1	4.8
Lung	97.2	2.4	96.7	2.2	96.7	2.2	97.2	2.4	97.2	2.4	97.2	2.4	97.2	2.4
GCM	59.5	19.9	51.6	7.4	53.7	10.3	57.9	10.9	57.9	11.8	60.0	13.4	62.1	14.2
Arcene	72.0	6.2	71.0	2.6	70.0	3.7	72.0	3.8	73.0	4.3	71.0	4.3	69.0	4.3
Madelon	60.5	8.0	61.7	2.0	60.8	3.4	61.3	4.8	60.3	5.9	60.1	5.6	60.5	6.2
Dorothea	92.9	6.3	94.3	3.0	93.3	4.0	93.4	5.0	93.0	5.3	92.9	5.3	92.9	5.3
Dexter	83.0	12.9	82.7	8.5	81.0	10.1	83.0	9.8	82.7	9.7	82.7	9.8	82.7	9.4
Gisette	94.1	30.7	86.2	2.7	88.5	6.4	90.8	10.8	91.6	15.9	92.3	16.5	92.6	17.1
Geom. Mean	*79.97*	*6.06*	*77.82*	*3.01*	*78.16*	*3.81*	*79.54*	*4.35*	*79.93*	*4.72*	*79.68*	*4.76*	*79.85*	*4.85*
Arith.Mean	*80.9*	*8.8*	*79.0*	*3.5*	*79.2*	*4.6*	*80.4*	*5.3*	*80.8*	*6.0*	*80.6*	*6.2*	*80.7*	*6.3*

Table A.3: Results using Naive Bayes classifier, IWSS$_r^2$ selection algorithm and CMIM-based re-ranking with block sizes B.

DataSet	**BARS**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	85.71	3.0	79.03	3.5	83.87	3.4	79.03	3.3	77.42	3.1	79.03	2.9	79.03	3.0
Leukemia	90.54	2.3	91.67	3.1	93.06	3.3	93.06	3.1	93.06	3.1	94.44	3.2	94.44	3.2
Lymphoma	73.67	6.1	71.88	8.3	73.96	7.8	79.17	9.2	78.13	9.1	77.08	9.1	79.17	9.5
DLBCL	76.00	2.4	85.11	3.1	85.11	3.3	74.47	3.4	74.47	3.6	80.85	3.2	76.60	3.3
Prostate	86.81	3.7	76.47	4.0	67.65	5.0	68.38	5.5	70.59	5.2	77.21	5.9	73.53	6.6
Lung	98.36	3.0	95.58	3.1	97.24	3.6	96.13	3.7	96.13	3.6	96.13	3.5	96.69	3.5
GCM	60.00	15.9	48.42	8.9	55.26	11.4	55.26	14.0	59.47	15.7	60.53	17.2	61.05	17.6
Arcene	74.00	4.9	76.00	4.6	81.00	4.6	82.00	6.0	78.00	6.5	83.00	7.3	85.00	6.7
Madelon	60.30	5.8	61.00	2.7	61.30	4.2	61.05	6.3	61.40	7.3	61.20	8.9	61.05	8.7
Dorothea	93.88	7.3	93.88	6.3	93.63	10.3	94.75	9.1	94.25	11.1	94.00	13.4	94.38	17.1
Dexter	82.67	12.8	77.00	6.1	84.67	12.8	82.67	15.5	83.33	15.5	83.67	15.8	84.67	15.2
Gisette	93.10	13.6	87.05	3.8	87.50	5.8	88.98	7.5	92.02	13.6	92.28	14.3	89.72	14.3
Geom. Mean	*80.29*	*5.4*	*77.33*	*4.4*	*79.26*	*5.6*	*78.52*	*6.3*	*78.94*	*6.8*	*80.77*	*7.2*	*80.41*	*7.39*
Arith. Mean	*81.25*	*6.7*	*78.59*	*4.8*	*80.35*	*6.3*	*79.58*	*7.2*	*79.86*	*8.1*	*81.62*	*8.7*	*81.28*	*9.06*

Table A.4: Results using Naive Bayes classifier, BARS selection algorithm and CMIM-based re-ranking with block sizes B.

DataSet	SFS		B=5		B=10		B=20		B=30		B=40		B=50	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	83.87	5.9	83.87	2.7	82.26	4.2	80.65	3.8	82.26	4.4	80.65	4.4	80.65	4.8
Leukemia	87.50	3.2	91.67	3.6	94.44	3.2	94.44	3.5	94.44	3.7	93.06	3.8	93.06	3.7
Lymphoma	83.33	7.1	76.04	9.0	76.04	11.5	82.29	9.2	81.25	10.7	80.21	9.2	79.17	9.6
DLBCL	80.85	3.6	91.49	4.4	91.49	4.3	91.49	4.6	91.49	4.5	91.49	4.2	89.36	4.2
Prostate	75.00	5.4	75.00	4.3	76.47	5.7	75.74	5.1	77.21	6.1	75.00	5.7	75.00	5.5
Lung	93.92	2.5	96.13	3.4	96.13	3.0	96.69	3.0	97.79	3.0	97.24	2.8	96.69	2.6
GCM	58.42	18.3	65.26	19.9	70.00	24.5	67.89	25.0	70.53	22.6	72.11	29.3	65.79	23.0
Arcene	68.00	4.6	74.00	6.8	72.00	8.4	75.00	7.0	74.00	7.3	73.00	8.1	72.00	7.7
Madelon	60.75	6.5	61.25	2.4	60.75	3.7	61.75	6.8	61.30	7.0	61.10	6.4	61.10	5.7
Dorothea	91.25	13.2	93.25	6.2	92.63	4.5	93.50	7.4	92.88	10.1	92.50	9.9	92.38	11.8
Dexter	76.00	13.8	80.00	12.0	83.00	13.5	83.67	12.5	83.67	13.0	84.33	12.8	80.33	10.8
Gisette	94.05	26.9	90.83	16.7	89.70	15.8	90.47	19.1	90.83	21.7	91.20	23.9	92.00	19.8
Geom. Mean	*78.51*	*7.1*	*80.79*	*6.0*	*81.34*	*6.7*	*82.07*	*7.2*	*82.43*	*7.8*	*81.96*	*7.8*	*80.68*	*7.4*
Arith. Mean	*79.41*	*9.3*	*81.57*	*7.6*	*82.08*	*8.5*	*82.80*	*8.9*	*83.14*	*9.5*	*82.66*	*10.0*	*81.46*	*9.1*

Table A.5: Results using Naive Bayes classifier, SFS selection algorithm and MIFS-based re-ranking with block sizes B.

DataSet	**IWSS^2**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	80.65	3.8	80.65	2.8	83.87	2.9	83.87	3.2	82.26	3.3	82.26	3.4	82.26	3.4
Leukemia	87.50	2.5	87.50	2.1	87.50	2.4	87.50	2.4	87.50	2.4	87.50	2.4	87.50	2.5
Lymphoma	76.04	8.8	71.88	7.0	72.92	7.3	72.92	8.2	72.92	8.1	76.04	8.2	77.08	8.3
DLBCL	85.11	1.9	89.36	1.5	87.23	1.6	85.11	1.7	85.11	1.7	85.11	1.7	85.11	1.7
Prostate	77.94	11.1	73.53	4.5	66.91	4.3	72.06	5.4	71.32	6.1	69.85	6.3	73.53	5.9
Lung	97.24	2.7	96.69	2.3	96.69	2.4	97.24	2.7	97.24	2.7	97.24	2.7	97.24	2.7
GCM	64.21	36.6	65.26	14.9	60.53	18.9	63.16	20.5	61.58	21.3	64.74	22.9	64.21	23.2
Arcene	70.00	13.4	73.00	4.4	73.00	7.4	73.00	7.6	72.00	7.3	71.00	7.7	65.00	8.4
Madelon	59.85	13.3	61.15	3.0	60.65	4.7	60.25	7.1	59.75	8.0	59.95	10.1	59.60	11.4
Dorothea	93.50	7.4	93.50	3.4	93.75	3.7	94.13	4.0	93.88	4.4	93.88	4.3	93.88	4.3
Dexter	81.00	19.6	84.00	10.7	83.33	12.5	82.67	12.4	81.00	13.1	79.33	13.6	79.67	14.4
Gisette	94.68	112.6	90.50	14.8	90.23	15.8	89.83	22.1	90.60	33.1	91.10	38.4	92.45	41.4
Geom. Mean	*79.81*	*9.4*	*79.80*	*4.5*	*78.77*	*5.2*	*79.31*	*5.9*	*78.71*	*6.3*	*79.01*	*6.7*	*78.88*	*6.9*
Arith. Mean	*80.64*	*19.5*	*80.58*	*6.0*	*79.72*	*7.0*	*80.14*	*8.1*	*79.60*	*9.3*	*79.83*	*10.1*	*79.79*	*10.6*

Table A.6: Results using Naive Bayes classifier, IWSS^2 selection algorithm and MIFS-based re-ranking with block sizes B.

DataSet	**IWSS^2_r**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	83.87	2.8	82.26	2.0	85.48	2.3	83.87	2.3	83.87	2.3	83.87	2.3	83.87	2.3
Leukemia	87.50	2.0	87.50	2.0	87.50	1.9	87.50	1.9	87.50	1.9	87.50	1.9	87.50	2.0
Lymphoma	80.21	5.9	72.92	5.7	78.13	5.1	81.25	5.0	80.21	5.4	82.29	5.6	78.13	5.6
DLBCL	80.85	1.8	87.23	1.5	82.98	1.6	80.85	1.7	80.85	1.7	80.85	1.7	80.85	1.7
Prostate	78.68	7.0	72.79	3.9	73.53	3.5	75.74	4.2	75.00	4.0	75.74	4.3	76.47	4.1
Lung	97.24	2.4	96.69	2.2	96.69	2.3	97.24	2.4	97.24	2.4	97.24	2.4	97.24	2.4
GCM	59.47	19.9	55.79	9.1	61.58	12.5	64.74	12.6	63.68	15.7	65.79	14.8	62.11	15.3
Arcene	72.00	6.2	70.00	3.3	71.00	3.9	70.00	3.9	72.00	4.6	72.00	5.3	70.00	4.9
Madelon	60.50	8.0	61.10	2.1	60.55	3.6	61.25	4.8	60.25	5.9	60.10	5.6	60.50	6.2
Dorothea	92.88	6.3	93.50	3.7	93.63	5.0	93.50	5.3	93.50	5.5	93.63	5.3	93.50	5.4
Dexter	83.00	12.9	84.00	9.0	82.67	10.0	82.33	8.9	83.67	9.2	83.67	9.4	82.67	9.5
Gisette	94.07	30.7	90.10	10.8	90.02	9.7	91.32	11.5	91.50	14.8	91.60	14.5	92.05	19.0
Geom. Mean	*79.97*	*6.1*	*78.46*	*3.7*	*79.48*	*4.1*	*80.05*	*4.4*	*79.99*	*4.7*	*80.45*	*4.8*	*79.58*	*4.9*
Arith. Mean	*80.85*	*8.8*	*79.49*	*4.6*	*80.31*	*5.1*	*80.80*	*5.4*	*80.77*	*6.1*	*81.19*	*6.1*	*80.41*	*6.5*

Table A.7: Results using Naive Bayes classifier, IWSS^2_r selection algorithm and MIFS-based re-ranking with block sizes B.

DataSet	**BARS**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	85.71	3.0	82.26	3.1	82.26	3.5	77.42	3.4	79.03	3.2	79.03	2.8	79.03	3.0
Leukemia	90.54	2.3	91.67	3.0	91.67	3.2	93.06	3.1	93.06	3.1	93.06	3.0	93.06	3.0
Lymphoma	73.67	6.1	73.96	6.8	75.00	7.9	79.17	9.2	79.17	8.2	77.08	8.3	76.04	8.6
DLBCL	76.00	2.4	82.98	2.8	93.62	3.4	76.60	3.4	78.72	3.4	82.98	3.3	82.98	3.4
Prostate	86.81	3.7	80.15	3.8	72.06	4.5	72.79	4.9	76.47	4.8	74.26	5.4	75.00	5.4
Lung	98.36	3.0	96.13	3.3	97.24	3.3	96.13	3.4	96.13	3.5	96.13	3.3	96.69	3.3
GCM	60.00	15.9	55.79	10.8	66.32	12.9	63.68	15.2	61.05	15.8	63.68	17.2	63.16	17.2
Arcene	74.00	4.9	82.00	4.5	80.00	5.2	80.00	6.4	79.00	6.6	80.00	6.3	81.00	5.9
Madelon	60.30	5.8	60.70	2.0	60.60	3.8	61.05	6.3	61.40	7.3	61.20	8.9	61.05	8.7
Dorothea	93.88	7.3	93.75	6.2	93.50	7.4	94.50	9.6	94.00	10.0	94.00	10.5	93.88	10.9
Dexter	82.67	12.8	77.67	7.3	79.33	11.1	81.67	13.5	84.33	12.6	83.33	13.5	83.33	13.7
Gisette	93.10	13.6	88.42	5.8	88.69	8.6	88.57	10.7	81.47	12.4	86.17	11.6	87.60	11.2
Geom. Mean	*80.29*	*5.4*	*79.50*	*4.4*	*80.89*	*5.5*	*79.63*	*6.4*	*79.57*	*6.5*	*80.19*	*6.6*	*80.31*	*6.6*
Arith. Mean	*81.25*	*6.7*	*80.46*	*5.0*	*81.69*	*6.2*	*80.39*	*7.4*	*80.32*	*7.6*	*80.91*	*7.8*	*81.07*	*7.9*

Table A.8: Results using Naive Bayes classifier, BARS selection algorithm and MIFS-based re-ranking with block sizes B.

DataSet	**SFS**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	83.87	5.9	85.48	2.1	85.48	2.8	80.65	4.4	82.26	3.9	80.65	3.9	80.65	4.3
Leukemia	87.50	3.2	88.89	2.9	95.83	3.4	94.44	3.2	94.44	3.6	93.06	3.8	93.06	3.7
Lymphoma	83.33	7.1	73.96	8.0	80.21	10.8	80.21	7.9	79.17	8.7	77.08	8.2	73.96	8.2
DLBCL	80.85	3.6	91.49	3.0	91.49	3.2	89.36	3.8	91.49	4.1	89.36	4.1	89.36	3.9
Prostate	75.00	5.4	76.47	3.8	77.21	4.5	75.00	3.6	72.79	4.1	74.26	4.0	74.26	4.0
Lung	93.92	2.5	95.58	2.8	96.13	2.8	96.69	3.0	97.79	3.0	97.24	2.8	96.69	2.6
GCM	58.42	18.3	50.53	13.2	51.58	10.5	51.05	11.5	57.37	13.8	53.16	12.6	55.79	18.8
Arcene	68.00	4.6	78.00	4.8	73.00	7.0	73.00	6.2	71.00	5.8	67.00	5.7	69.00	4.9
Madelon	60.75	6.5	61.25	2.4	60.75	3.7	61.75	6.8	61.30	7.0	61.10	6.4	61.10	5.7
Dorothea	91.25	13.2	93.25	4.5	92.75	4.2	93.38	4.6	93.00	6.4	92.88	5.3	92.88	5.3
Dexter	76.00	13.8	75.00	6.4	80.67	7.7	81.33	8.5	84.00	11.0	83.00	12.2	80.67	9.9
Gisette	94.05	26.9	88.00	9.0	88.10	11.2	89.20	14.1	88.73	13.9	89.75	13.1	90.23	22.2
Geom. Mean	*78.51*	*7.1*	*78.63*	*4.5*	*79.86*	*5.2*	*79.29*	*5.7*	*80.06*	*6.2*	*78.67*	*6.0*	*78.72*	*6.2*
Arith. Mean	*79.41*	*9.3*	*79.82*	*5.2*	*81.10*	*6.0*	*80.50*	*6.5*	*81.11*	*7.1*	*79.88*	*6.8*	*79.80*	*7.8*

Table A.9: Results using Naive Bayes classifier, SFS selection algorithm and MRMR-based re-ranking with block sizes B.

DataSet	IWSS^2		B=5		B=10		B=20		B=30		B=40		B=50	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	80.65	3.8	80.65	2.7	83.87	2.9	83.87	3.0	83.87	3.1	83.87	3.2	83.87	3.2
Leukemia	87.50	2.5	87.50	1.8	87.50	2.1	87.50	2.1	87.50	2.2	87.50	2.3	87.50	2.4
Lymphoma	76.04	8.8	64.58	5.1	72.92	5.6	69.79	6.5	70.83	6.8	73.96	7.4	75.00	7.8
DLBCL	85.11	1.9	89.36	1.5	87.23	1.6	85.11	1.7	85.11	1.7	85.11	1.7	85.11	1.7
Prostate	77.94	11.1	70.59	3.2	70.59	3.9	74.26	4.9	73.53	5.9	72.79	6.1	73.53	6.1
Lung	97.24	2.7	96.13	2.1	96.13	2.2	97.24	2.7	97.24	2.7	97.24	2.7	97.24	2.7
GCM	64.21	36.6	49.47	8.9	48.95	8.7	55.79	13.1	51.05	14.9	52.63	17.9	54.21	17.9
Arcene	70.00	13.4	74.00	3.1	73.00	4.9	69.00	5.9	73.00	6.0	74.00	6.5	77.00	7.3
Madelon	59.85	13.3	61.15	3.0	60.65	4.7	60.25	7.1	59.75	8.0	59.95	10.1	59.60	11.4
Dorothea	93.50	7.4	93.50	2.4	93.88	2.9	94.25	3.2	94.00	3.8	94.00	3.8	94.00	3.9
Dexter	81.00	19.6	72.00	5.1	77.67	5.3	83.67	9.9	82.67	11.6	81.00	13.2	79.67	14.3
Gisette	94.68	112.6	88.48	6.0	88.00	8.2	86.68	13.3	87.23	19.2	88.28	19.1	89.85	21.8
Geom. Mean	*79.81*	*9.4*	*75.95*	*3.3*	*77.09*	*3.9*	*77.90*	*4.9*	*77.61*	*5.5*	*78.08*	*5.9*	*78.65*	*6.2*
Arith. Mean	*80.64*	*19.5*	*77.28*	*3.7*	*78.37*	*4.4*	*78.95*	*6.1*	*78.82*	*7.2*	*79.19*	*7.8*	*79.71*	*8.4*

Table A.10: Results using Naive Bayes classifier, IWSS^2 selection algorithm and MRMR-based re-ranking with block sizes B.

DataSet	IWSS$_r^2$		B=5		B=10		B=20		B=30		B=40		B=50	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	83.87	2.8	80.65	2.0	80.65	2.3	80.65	2.3	82.26	2.3	85.48	2.3	85.48	2.3
Leukemia	87.50	2.0	87.50	1.8	87.50	1.9	87.50	1.9	87.50	1.9	87.50	1.9	87.50	2.0
Lymphoma	80.21	5.9	67.71	4.1	72.92	4.8	79.17	4.8	78.13	4.8	80.21	5.4	77.08	5.5
DLBCL	80.85	1.8	87.23	1.5	82.98	1.6	80.85	1.7	80.85	1.7	80.85	1.7	80.85	1.7
Prostate	78.68	7.0	72.79	2.9	74.26	3.1	77.94	3.6	77.94	3.5	75.74	3.6	75.00	3.6
Lung	97.24	2.4	96.13	2.1	96.13	2.1	97.24	2.4	97.24	2.4	97.24	2.4	97.24	2.4
GCM	59.47	19.9	48.95	6.5	47.37	8.3	54.21	9.1	53.68	9.3	49.47	10.0	48.42	10.5
Arcene	72.00	6.2	72.00	2.5	70.00	4.0	71.00	4.5	71.00	4.8	72.00	4.8	73.00	4.8
Madelon	60.50	8.0	61.10	2.1	60.55	3.6	61.25	4.8	60.25	5.9	60.10	5.6	60.50	6.2
Dorothea	92.88	6.3	93.75	3.8	92.63	3.9	93.50	3.9	93.38	4.0	93.25	4.0	93.25	4.0
Dexter	83.00	12.9	73.00	5.0	79.33	4.9	82.00	7.7	81.33	8.5	81.00	8.9	81.67	9.2
Gisette	94.07	30.7	88.28	5.8	88.38	7.1	89.68	6.5	88.67	7.3	88.40	9.8	89.60	13.4
Geom. Mean	*79.97*	*6.1*	*76.14*	*3.0*	*76.43*	*3.5*	*78.59*	*3.9*	*78.32*	*4.1*	*78.04*	*4.2*	*77.86*	*4.4*
Arith. Mean	*80.85*	*8.8*	*77.42*	*3.3*	*77.72*	*4.0*	*79.58*	*4.4*	*79.35*	*4.7*	*79.27*	*5.0*	*79.13*	*5.5*

Table A.11: Results using Naive Bayes classifier, IWSS$_r^2$ selection algorithm and MRMR-based re-ranking with block sizes B.

DataSet	**BARS**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	85.71	3.0	82.26	2.9	82.26	3.1	80.65	2.8	79.03	2.8	79.03	2.6	79.03	2.6
Leukemia	90.54	2.3	91.67	2.2	91.67	3.0	93.06	2.9	93.06	2.9	93.06	2.8	93.06	2.8
Lymphoma	73.67	6.1	69.79	5.5	71.88	5.6	70.83	7.3	72.92	6.4	72.92	6.8	70.83	7.1
DLBCL	76.00	2.4	85.11	2.4	85.11	2.8	76.60	2.7	80.85	2.8	80.85	2.9	80.85	2.9
Prostate	86.81	3.7	75.00	3.6	75.00	3.9	73.53	3.7	73.53	4.2	75.74	4.2	75.00	4.6
Lung	98.36	3.0	96.69	2.6	97.24	3.3	96.13	3.4	96.13	3.5	96.13	3.3	96.69	3.3
GCM	60.00	15.9	48.42	6.9	46.84	7.2	47.37	8.6	51.58	9.2	49.47	10.6	50.53	10.8
Arcene	74.00	4.9	77.00	4.8	79.00	4.2	81.00	5.2	77.00	5.0	80.00	4.5	82.00	4.1
Madelon	60.30	5.8	60.70	2.0	60.60	3.8	61.05	6.3	61.40	7.3	61.20	8.9	61.05	8.7
Dorothea	93.88	7.3	93.38	5.1	93.75	5.3	94.63	6.1	94.13	6.7	94.13	6.8	94.00	7.8
Dexter	82.67	12.8	75.33	6.2	79.33	8.7	76.67	6.2	83.33	9.3	82.67	13.1	83.00	13.4
Gisette	93.10	13.6	88.55	6.3	87.77	8.1	88.47	6.5	88.70	6.5	88.70	6.7	88.70	6.7
Geom. Mean	*80.29*	*5.4*	*77.34*	*3.8*	*77.81*	*4.6*	*76.97*	*4.8*	*78.16*	*5.1*	*78.26*	*5.3*	*78.34*	*5.4*
Arith. Mean	*81.25*	*6.7*	*78.66*	*4.2*	*79.20*	*4.9*	*78.33*	*5.1*	*79.30*	*5.6*	*79.49*	*6.1*	*79.56*	*6.2*

Table A.12: Results using Naive Bayes classifier, BARS selection algorithm and MRMR-based re-ranking with block sizes B.

A. RE-RANKING CRITERIA

Appendix B

CMIM Re-ranking Using Different Classifiers for IWSS and IWSS with replacement

B.1 IWSS and CMIM Criterion

DataSet	**IWSS^2**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	80.6	3.8	80.6	2.8	83.9	3.0	82.3	3.2	82.3	3.3	82.3	3.3	82.3	3.3
Leukemia	87.5	2.5	87.5	2.0	87.5	2.4	87.5	2.4	87.5	2.4	87.5	2.4	87.5	2.5
Lymphoma	76.0	8.8	66.7	6.3	75.0	7.6	76.0	7.9	77.1	8.0	75.0	8.1	77.1	8.2
DLBCL	85.1	1.9	89.4	1.5	87.2	1.6	85.1	1.7	85.1	1.7	85.1	1.7	85.1	1.7
Prostate	77.9	11.1	72.1	4.1	74.3	4.0	77.9	5.6	74.3	7.3	72.1	7.8	74.3	8.0
Lung	97.2	2.7	96.7	2.2	96.7	2.4	97.2	2.7	97.2	2.7	97.2	2.7	97.2	2.7
GCM	64.2	36.6	54.2	12.3	60.0	19.8	62.1	21.4	65.3	22.5	64.2	24.4	64.7	27.4
Arcene	70.0	13.4	70.0	3.5	68.0	5.1	70.0	6.8	70.0	7.0	70.0	7.8	69.0	7.8
Madelon	59.9	13.3	61.3	2.7	60.9	4.8	60.3	7.1	59.8	8.0	60.0	10.1	59.6	11.4
Dorothea	93.5	7.4	93.9	2.8	94.1	3.6	94.4	3.8	94.0	4.3	93.9	4.3	93.8	4.5
Dexter	81.0	19.6	81.7	11.9	83.7	13.1	83.7	14.8	81.3	15.7	83.0	15.2	80.7	14.9
Gisette	94.7	112.6	88.7	18.3	92.3	41.3	93.7	62.6	93.9	69.5	94.4	82.0	94.1	77.2
Geom. Mean	*79.81*	*9.45*	*77.41*	*4.24*	*79.34*	*5.52*	*79.97*	*6.50*	*79.81*	*6.94*	*79.51*	*7.32*	*79.59*	*7.49*
Arith. Mean	*80.6*	*19.5*	*78.5*	*5.9*	*80.3*	*9.1*	*80.9*	*11.7*	*80.6*	*12.7*	*80.4*	*14.2*	*80.4*	*14.1*

Table B.1: NB classifier, IWSS^2 and CMIM-based re-ranking with block sizes B.

B.2 IWSS with replacement and CMIM Criterion

DataSet	**IWSS^2**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	79.03	2.7	82.26	1.7	82.26	1.7	82.26	1.9	82.26	2.0	82.26	2.0	80.65	2.3
Leukemia	83.33	1.1	83.33	1.0	83.33	1.0	83.33	1.0	83.33	1.1	83.33	1.1	83.33	1.1
Lymphoma	75.00	8.9	64.58	5.8	65.63	6.4	78.13	7.8	78.13	7.9	76.04	8.1	76.04	8.2
DLBCL	76.60	1.5	76.60	1.4	76.60	1.4	76.60	1.4	76.60	1.4	76.60	1.4	76.60	1.5
Prostate	88.24	4.9	88.97	3.9	89.71	4.0	89.71	4.1	91.18	4.4	91.18	4.3	91.18	4.3
LungCancer	95.03	1.3	95.58	1.1	95.58	1.1	95.58	1.2	95.58	1.2	95.58	1.2	95.58	1.2
GCM	45.26	23.4	45.26	8.5	48.42	10.9	45.79	13.6	48.42	15.3	44.74	14.3	45.26	14.1
Arcene	82.00	7.5	78.00	4.0	79.00	5.1	79.00	5.6	78.00	5.9	78.00	6.0	78.00	6.3
Madelon	77.25	23.1	75.60	8.0	76.15	9.6	75.60	13.1	76.70	13.8	75.35	14.9	76.05	19.8
Dorothea	91.63	9.7	92.38	2.5	92.25	3.5	91.63	5.2	91.75	5.3	91.75	5.5	91.88	5.7
Dexter	81.33	13.2	76.33	5.4	81.67	8.4	81.67	8.7	80.67	9.4	82.33	9.3	82.67	9.4
Gisette	93.70	67.8	92.95	21.2	93.48	28.1	93.72	37.4	93.70	42.3	93.30	49.2	94.08	52.9
Geom. Mean	*79.48*	*6.7*	*77.97*	*3.6*	*79.17*	*4.2*	*79.88*	*4.9*	*80.29*	*5.1*	*79.58*	*5.2*	*79.68*	*5.5*
Arith. Mean	*80.70*	*13.8*	*79.32*	*5.4*	*80.34*	*6.8*	*81.08*	*8.4*	*81.36*	*9.2*	*80.87*	*9.8*	*80.94*	*10.6*

Table B.2: c4.5 classifier, IWSS^2 and CMIM-based re-ranking with block sizes B.

DataSet	**IWSS^2**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	82.26	6.3	83.87	4.0	80.65	4.5	77.42	5.1	80.65	4.9	80.65	5.2	80.65	5.0
Leukemia	88.89	2.8	84.72	2.3	83.33	2.4	84.72	2.5	86.11	2.8	86.11	2.8	87.50	2.8
Lymphoma	81.25	12.5	75.00	8.0	79.17	10.2	81.25	10.9	81.25	11.0	85.42	11.3	85.42	11.5
DLBCL	85.11	3.5	80.85	2.8	78.72	3.0	76.60	3.2	78.72	3.2	80.85	3.3	80.85	3.3
Prostate	86.03	8.6	86.03	5.0	89.71	5.9	88.97	6.1	90.44	6.7	88.97	6.7	88.97	6.7
LungCancer	96.13	2.7	95.03	2.6	95.58	2.6	96.13	2.6	96.13	2.6	96.13	2.7	96.13	2.7
GCM	65.26	34.1	49.47	9.8	50.00	11.6	55.26	17.8	59.47	21.4	57.89	20.2	57.89	23.6
Arcene	76.00	13.2	78.00	5.9	79.00	6.4	76.00	8.2	77.00	8.2	79.00	8.7	76.00	8.6
Madelon	88.00	11.7	85.65	8.4	85.45	7.7	85.40	7.7	86.75	9.1	87.00	9.6	87.00	9.6
Dorothea	91.88	18.2	93.50	4.3	93.38	6.0	93.13	7.5	92.75	8.4	92.50	8.7	92.63	9.9
Dexter	83.33	24.6	81.67	12.0	81.67	15.2	81.67	15.6	81.67	15.9	82.67	16.7	83.67	16.9
Gisette	95.97	100.4	90.37	18.5	93.77	38.0	95.95	65.4	95.95	70.0	95.88	72.0	96.10	81.3
Geom. Mean	*84.58*	*11.3*	*81.05*	*5.7*	*81.56*	*6.8*	*81.92*	*7.9*	*83.29*	*8.4*	*83.78*	*8.6*	*83.74*	*8.9*
Arith. Mean	*85.01*	*19.9*	*82.01*	*7.0*	*82.53*	*9.5*	*82.71*	*12.7*	*83.91*	*13.7*	*84.42*	*14.0*	*84.40*	*15.2*

Table B.3: ibK classifier, IWSS^2 and CMIM-based re-ranking with block sizes B.

DataSet	**IWSS^2**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	67.74	1.1	69.35	1.0	69.35	1.0	69.35	1.0	69.35	1.0	69.35	1.0	69.35	1.0
Leukemia	65.28	1.0	65.28	1.0	65.28	1.0	65.28	1.0	65.28	1.0	65.28	1.0	65.28	1.0
Lymphoma	82.29	11.1	80.21	8.2	81.25	8.1	80.21	8.8	80.21	8.9	80.21	9.0	81.25	9.8
DLBCL	87.23	1.9	89.36	1.5	89.36	1.5	87.23	1.6	85.11	1.8	87.23	1.9	87.23	1.9
Prostate	73.53	1.2	76.47	1.0	76.47	1.0	76.47	1.0	76.47	1.0	76.47	1.0	76.47	1.0
LungCancer	82.32	1.0	82.32	1.0	82.32	1.0	82.32	1.0	82.32	1.0	82.32	1.0	82.32	1.0
GCM	15.79	1.7	14.74	1.3	14.74	1.3	14.74	1.4	14.74	1.4	14.74	1.4	14.74	1.4
Arcene	61	3.8	64.00	1.5	63.00	1.7	63.00	2.1	63.00	2.2	62.00	2.6	63.00	2.7
Madelon	57.85	7.7	56.40	1.9	56.30	1.8	56.10	3.4	56.60	3.7	56.65	3.8	56.65	3.9
Dorothea	93.00	6.1	93.38	3.3	93.25	5.1	92.75	5.5	92.88	5.6	93.00	5.5	93.13	5.5
Dexter	80.33	21.2	71.33	4.0	80.67	9.0	80.00	12.0	82.33	12.1	81.33	12.0	81.33	12.1
Gisette	89.03	28.5	87.03	7.5	88.33	9.2	88.48	10.1	88.72	13.0	88.55	11.8	88.72	11.9
Geom. Mean	*66.23*	*3.62*	*65.56*	*2.02*	*66.28*	*2.29*	*65.99*	*2.60*	*66.08*	*2.72*	*66.07*	*2.75*	*66.24*	*2.78*
Arith. Mean	*71.28*	*7.19*	*70.82*	*2.77*	*71.69*	*3.48*	*71.33*	*4.08*	*71.42*	*4.39*	*71.43*	*4.33*	*71.62*	*4.43*

Table B.4: SVM classifier, IWSS^2 and CMIM-based re-ranking with block sizes B.

DataSet	**IWSS^2**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	62.90	4.2	67.74	2.2	67.74	2.5	66.13	2.9	64.52	3.2	64.52	3.2	64.52	3.1
Leukemia	87.50	2.6	84.02	2.0	84.72	2.1	86.11	2.2	86.11	2.4	86.11	2.4	86.11	2.5
Lymphoma	81.25	11.3	80.21	7.6	80.21	8.0	80.21	8.0	72.92	8.1	76.04	8.8	79.17	9.2
DLBCL	78.72	2.0	82.98	1.3	80.85	1.5	76.60	1.7	76.60	1.7	76.60	1.7	74.47	1.8
Prostate	89.71	7.0	87.00	4.1	88.97	4.2	90.44	4.5	91.91	4.8	91.18	4.5	92.65	4.7
LungCancer	97.79	2.4	96.69	2.1	96.69	2.2	96.69	2.2	96.69	2.2	96.69	2.2	96.69	2.2
GCM	55.79	33.4	50.50	13.2	52.63	15.1	54.74	17.4	58.42	19.6	53.68	17.8	59.47	21.1
Arcene	83.00	12.9	78.00	4.3	79.00	6.4	80.00	7.2	78.00	8.6	77.00	8.9	76.00	9.4
Madelon	75.85	12.4	75.20	8.5	75.15	9.4	75.75	10.4	76.40	11.0	76.75	11.2	76.40	11.5
Dorothea	–	–	92.75	3.7	92.25	6.0	92.25	6.7	92.25	7.4	92.38	7.9	92.88	7.8
Dexter	8.00	2149.4	86.24	12.5	86.33	13.1	84.67	12.5	88.00	13.5	88.33	13.3	87.33	13.2
Gisette	–	–	93.40	18.9	93.53	21.0	94.23	26.9	94.63	29.7	94.25	29.9	95.28	39.1
Geom. Mean	–	–	*80.17*	*4.8*	*80.54*	*5.6*	*80.56*	*6.1*	*80.47*	*6.5*	*80.08*	*6.6*	*80.87*	*6.9*
Arith. Mean	–	–	*81.23*	*6.7*	*81.51*	*7.6*	*81.48*	*8.6*	*81.37*	*9.4*	*81.13*	*9.3*	*81.75*	*10.5*

Table B.5: MLP classifier, IWSS^2 and CMIM-based re-ranking with block sizes B.

DataSet	**IWSS^2**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	87.10	3.0	88.71	2.8	87.10	2.9	87.10	2.9	87.10	2.9	87.10	2.9	87.10	3.0
Leukemia	91.67	3.2	91.67	2.7	91.67	2.8	91.67	2.9	91.67	2.9	91.67	2.9	91.67	2.9
Lymphoma	72.92	10.4	64.58	7.0	72.92	8.0	72.92	9.7	79.17	10.0	80.21	9.5	79.17	9.4
DLBCL	82.98	2.9	85.11	2.5	82.98	2.7	82.98	2.8	82.98	2.8	82.98	2.8	82.98	2.8
Prostate	90.44	6.3	90.44	4.4	93.38	4.9	94.12	5.1	92.65	5.8	93.38	6.0	94.12	6.1
LungCancer	96.13	2.9	96.13	2.5	96.69	2.6	96.69	2.6	96.69	2.6	96.69	2.6	96.69	2.6
GCM	55.79	34.8	52.11	14.2	51.05	17.0	51.58	19.8	54.74	25.0	58.95	24.4	60.53	26.6
Arcene	78.00	7.6	84.00	4.4	82.00	5.1	80.00	6.2	80.00	6.7	82.00	6.8	81.00	7.0
Madelon	63.60	4.7	63.70	4.8	63.60	4.7	63.60	4.7	63.60	4.7	63.60	4.7	63.60	4.7
Dorothea	93.88	15.6	92.50	6.8	93.00	7.5	93.13	8.7	93.00	9.0	93.25	9.7	93.13	9.8
Dexter	84.00	11.0	75.33	4.2	80.33	8.7	81.67	11.3	84.00	11.0	84.00	11.0	84.00	11.0
Gisette	96.25	112.8	92.43	24.0	93.90	40.9	95.18	68.2	95.62	75.7	95.43	80.9	95.87	86.9
Geom. Mean	*81.68*	*8.42*	*80.10*	*5.1*	*81.14*	*6.1*	*81.31*	*6.9*	*82.38*	*7.2*	*83.21*	*7.3*	*83.30*	*7.5*
Arith. Mean	*82.73*	*17.93*	*81.39*	*6.7*	*82.38*	*9.0*	*82.55*	*12.1*	*83.43*	*13.3*	*84.10*	*13.7*	*84.15*	*14.4*

Table B.6: TAN classifier, IWSS^2 and CMIM-based re-ranking with block sizes B.

DataSet	**IWSS^2**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	87.10	3.1	87.10	2.9	87.10	3.1	87.10	3.1	87.10	3.1	87.10	3.1	87.10	3.1
Leukemia	88.89	1.9	88.89	1.8	88.89	1.8	88.89	1.8	88.89	1.9	88.89	1.9	88.89	1.9
Lymphoma	86.46	8.3	84.38	7.0	87.50	7.8	88.54	8.3	87.50	8.3	87.50	8.3	86.46	8.3
DLBCL	80.85	1.8	80.85	1.5	80.85	1.5	80.85	1.5	80.85	1.6	80.85	1.7	80.85	1.8
Prostate	91.18	5.2	91.18	4.7	90.44	4.9	91.18	5.2	90.44	5.1	90.44	5.1	90.44	5.0
LungCancer	96.13	1.9	96.13	1.5	96.13	1.9	96.13	1.9	96.13	1.9	96.13	1.9	96.13	1.9
GCM	62.11	35.2	67.37	14.9	70.00	18.6	70.00	19.5	68.95	23.1	68.42	25.7	69.47	27.2
Arcene	93.00	6.0	92.00	4.5	93.00	5.3	93.00	5.8	93.00	5.8	92.00	5.9	92.00	5.9
Madelon	67.60	4.6	67.65	4.7	67.60	4.6	67.60	4.6	67.60	4.6	67.60	4.6	67.60	4.6
Dorothea	93.00	10.1	93.25	3.4	93.50	5.1	92.50	5.9	92.88	5.0	93.25	4.8	93.25	4.8
Dexter	86.67	13.4	77.67	5.9	86.00	12.2	86.00	13.1	85.67	12.7	86.67	13.8	87.00	13.5
Gisette	96.17	116.8	93.10	32.6	93.72	45.3	95.85	79.3	95.53	73.7	95.80	83.5	95.92	87.2
Geom. Mean	*84.88*	*7.5*	*84.41*	*4.6*	*85.74*	*5.5*	*85.97*	*6.0*	*85.70*	*6.0*	*85.70*	*6.2*	*85.76*	*6.2*
Arith. Mean	*85.64*	*18.7*	*84.96*	*7.1*	*86.23*	*9.3*	*86.47*	*12.5*	*86.21*	*12.2*	*86.22*	*13.4*	*86.26*	*13.8*

Table B.7: AODE classifier, IWSS^2 and CMIM-based re-ranking with block sizes B.

DataSet	**IWSS^2_r**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	83.9	2.8	82.3	2.2	85.5	2.2	83.9	2.4	83.9	2.4	83.9	2.3	83.9	2.3
Leukemia	87.5	2.0	87.5	1.9	87.5	1.9	87.5	1.9	87.5	1.9	87.5	1.9	87.5	2.0
Lymphoma	80.2	5.9	72.9	4.7	75.0	5.6	80.2	5.6	81.3	5.7	78.1	5.9	76.0	5.9
DLBCL	80.9	1.8	87.2	1.5	83.0	1.6	80.9	1.7	80.9	1.7	80.9	1.7	80.9	1.7
Prostate	78.7	7.0	73.5	3.2	75.7	3.4	77.2	4.2	80.9	4.7	80.1	4.7	83.1	4.8
Lung	97.2	2.4	96.7	2.2	96.7	2.2	97.2	2.4	97.2	2.4	97.2	2.4	97.2	2.4
GCM	59.5	19.9	51.6	7.4	53.7	10.3	57.9	10.9	57.9	11.8	60.0	13.4	62.1	14.2
Arcene	72.0	6.2	71.0	2.6	70.0	3.7	72.0	3.8	73.0	4.3	71.0	4.3	69.0	4.3
Madelon	60.5	8.0	61.7	2.0	60.8	3.4	61.3	4.8	60.3	5.9	60.1	5.6	60.5	6.2
Dorothea	92.9	6.3	94.3	3.0	93.3	4.0	93.4	5.0	93.0	5.3	92.9	5.3	92.9	5.3
Dexter	83.0	12.9	82.7	8.5	81.0	10.1	83.0	9.8	82.7	9.7	82.7	9.8	82.7	9.4
Gisette	94.1	30.7	86.2	2.7	88.5	6.4	90.8	10.8	91.6	15.9	92.3	16.5	92.6	17.1
Geom. Mean	*79.97*	*6.06*	*77.82*	*3.01*	*78.16*	*3.81*	*79.54*	*4.35*	*79.93*	*4.72*	*79.68*	*4.76*	*79.85*	*4.85*
Arith.Mean	*80.9*	8.8	*79.0*	*3.5*	*79.2*	*4.6*	*80.4*	*5.3*	*80.8*	*6.0*	*80.6*	*6.2*	*80.7*	*6.3*

Table B.8: NB classifier, IWSS^2_r and CMIM-based re-ranking with block sizes B.

DataSet	**IWSS^2_r**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	79.03	2.3	80.65	1.7	82.26	1.7	82.26	2.0	80.65	1.9	80.65	1.9	79.03	2.1
Leukemia	84.72	1.1	84.72	1.0	84.72	1.0	84.72	1.0	84.72	1.1	84.72	1.1	84.72	1.1
Lymphoma	77.08	6.4	65.63	4.9	65.63	5.0	75.00	5.8	76.04	5.7	75.00	5.9	75.00	6.0
DLBCL	82.98	1.3	80.85	1.2	78.72	1.2	78.72	1.3	78.72	1.3	78.72	1.3	78.72	1.3
Prostate	79.41	4.5	87.50	3.5	86.76	3.4	82.35	3.6	82.35	3.6	82.35	3.7	82.35	3.7
LungCancer	93.92	1.3	94.48	1.1	94.48	1.1	94.48	1.2	94.48	1.2	94.48	1.2	94.48	1.2
GCM	41.58	15.6	44.21	5.1	43.68	5.6	43.68	7.5	44.21	7.3	49.47	9.1	46.84	8.9
Arcene	80.00	5.4	77.00	2.5	79.00	3.4	82.00	4.1	82.00	4.4	82.00	4.7	84.00	4.8
Madelon	77.90	13.4	75.70	6.4	77.25	7.0	76.95	7.8	77.05	8.6	76.85	9.3	78.95	10.9
Dorothea	91.88	8.5	92.63	2.6	92.75	3.4	92.25	5.2	92.38	5.2	92.38	5.3	92.38	5.3
Dexter	79.67	13.8	75.33	4.2	79.33	6.4	77.33	6.8	79.33	6.9	79.33	6.8	80.00	6.9
Gisette	93.75	35.8	91.87	10.2	92.20	13.0	93.33	17.5	93.50	21.5	93.17	20.3	93.57	24.2
Geom. Mean	*78.74*	*5.4*	*77.84*	*2.9*	*78.33*	*3.3*	*78.96*	*3.8*	*79.20*	*3.9*	*79.81*	*4.1*	*79.74*	*4.2*
Arith. Mean	*80.16*	*9.1*	*79.21*	*3.7*	*79.73*	*4.4*	*80.26*	*5.3*	*80.45*	*5.7*	*80.76*	*5.9*	*80.84*	*6.4*

Table B.9: c4.5 classifier, IWSS^2_r and CMIM-based re-ranking with block sizes B.

DataSet	**IWSS$_r^2$**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	77.42	4.9	82.26	3.3	82.26	3.7	79.03	4.1	70.97	3.8	74.19	4.0	79.03	4.0
Leukemia	87.50	2.2	88.89	2.2	87.50	2.2	87.50	2.2	87.50	2.2	87.50	2.2	87.50	2.2
Lymphoma	78.13	7.7	80.21	5.5	76.04	5.7	82.29	6.3	79.17	7.0	77.08	7.0	77.08	7.0
DLBCL	85.11	2.5	80.85	2.4	80.85	2.4	80.85	2.4	80.85	2.5	80.85	2.4	80.85	2.4
Prostate	88.97	5.3	89.71	3.5	87.50	4.3	86.76	4.4	87.50	4.3	86.76	4.7	88.24	4.6
LungCancer	96.69	2.5	96.13	2.3	96.69	2.4	96.69	2.5	96.69	2.5	96.69	2.5	96.69	2.5
GCM	55.79	20.5	41.58	5.5	48.42	7.8	48.95	8.6	51.05	10.9	58.42	11.4	55.79	13.3
Arcene	72.00	6.9	79.00	4.0	76.00	4.9	75.00	5.2	78.00	6.0	74.00	6.3	71.00	6.2
Madelon	87.85	8.1	86.25	6.4	85.00	6.4	85.40	6.4	86.50	7.1	87.20	7.2	87.20	7.2
Dorothea	93.13	15.9	93.88	4.4	93.63	6.5	93.88	6.6	92.88	7.8	93.38	8.7	92.63	9.2
Dexter	80.00	20.1	85.33	9.7	77.00	11.9	82.33	12.7	80.67	13.5	80.67	13.9	81.00	13.6
Gisette	–	–	89.35	9.8	91.80	17.2	94.40	26.1	95.08	35.7	95.08	36.6	95.30	41.8
Geom. Mean	*81.25*	*6.6*	*81.26*	*4.3*	*80.81*	*5.2*	*81.69*	*5.6*	*81.23*	*6.1*	*81.94*	*6.3*	*81.88*	*6.5*
Arith. Mean	*82.05*	*8.8*	*82.79*	*4.9*	*81.89*	*6.3*	*82.76*	*7.3*	*82.24*	*8.6*	*82.65*	*8.9*	*82.69*	*9.5*

Table B.10: ibK classifier, IWSS$_r^2$ and CMIM-based re-ranking with block sizes B.

DataSet	**IWSS$_r^2$**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	64.52	1.0	67.74	1.0	66.13	1.0	62.90	1.0	62.90	1.0	59.68	1.0	59.68	1.0
Leukemia	70.83	1.0	65.28	1.0	65.28	1.0	65.28	1.0	65.28	1.0	65.28	1.0	65.28	1.0
Lymphoma	81.25	7.3	80.21	6.5	81.25	6.9	80.21	6.7	79.17	6.7	79.17	6.9	81.25	6.6
DLBCL	80.85	1.9	87.23	1.5	82.98	1.5	80.85	1.6	80.85	1.8	82.98	1.9	82.98	1.9
Prostate	72.06	1.0	72.79	1.0	71.32	1.0	72.06	1.0	72.06	1.0	72.06	1.0	72.06	1.0
LungCancer	87.29	1.0	83.98	1.0	88.95	1.0	87.85	1.0	87.29	1.0	87.29	1.0	87.29	1.0
GCM	20.53	1.1	16.84	1.1	17.37	1.1	16.32	1.1	16.84	1.1	17.89	1.1	17.37	1.1
Arcene	60.00	5.6	67.00	1.4	69.00	1.6	67.00	1.7	63.00	2.3	65.00	2.1	62.00	2.3
Madelon	57.65	5.8	57.45	1.3	57.10	1.5	57.35	2.4	57.35	2.4	57.35	2.7	57.35	2.7
Dorothea	92.50	5.1	93.75	3.4	92.88	4.6	93.38	4.6	93.00	5.0	93.00	5.0	93.00	5.0
Dexter	82.00	18.5	71.33	3.8	81.00	8.3	82.00	10.0	80.00	10.5	80.67	10.4	81.33	10.4
Gisette	–	–	87.85	4.8	88.62	5.4	89.47	6.9	89.63	7.3	89.88	7.7	89.95	7.9
Geom. Mean	–	–	*66.29*	*1.81*	*67.16*	*2.05*	*66.32*	*2.23*	*65.90*	*2.35*	*66.32*	*2.38*	*66.09*	*2.39*
Arith. Mean	–	–	*70.95*	*2.32*	*71.82*	*2.91*	*71.22*	*3.25*	*70.61*	*3.43*	*70.85*	*3.48*	*70.79*	*3.49*

Table B.11: SVM classifier, IWSS$_r^2$ and CMIM-based re-ranking with block sizes B.

DataSet	**IWSS$_r^2$**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	72.58	3.0	70.97	2.0	70.97	2.1	69.35	2.5	69.35	2.6	67.74	2.6	69.35	2.5
Leukemia	87.50	2.1	86.11	1.7	86.11	1.8	86.11	1.8	86.11	1.9	86.11	1.9	86.11	2.0
Lymphoma	83.33	7.5	73.96	5.0	68.75	4.6	78.13	5.8	83.33	5.8	83.33	6.3	80.21	6.4
DLBCL	78.72	2.0	82.98	1.3	80.85	1.5	76.60	1.7	76.60	1.7	76.60	1.7	74.47	1.8
Prostate	86.03	4.7	91.18	2.9	90.44	3.4	91.91	4.0	91.18	3.9	91.18	3.8	90.44	3.6
LungCancer	97.24	2.2	96.13	1.8	96.13	1.8	96.13	1.8	96.13	1.8	96.13	1.8	96.13	1.8
GCM	–	–	46.84	6.7	43.68	7.1	52.11	7.7	60.00	9.5	52.11	8.8	54.21	9.7
Arcene	74.00	7.0	78.00	3.9	79.00	4.2	77.00	5.3	78.00	6.0	80.00	5.6	79.00	5.4
Madelon	74.25	7.2	72.60	6.0	73.55	6.6	73.95	6.4	74.10	6.9	75.45	7.1	75.10	7.3
Dorothea	–	–	93.88	3.1	93.75	4.3	94.25	5.5	93.63	5.6	93.25	6.3	93.25	6.9
Dexter	–	–	82.00	7.4	84.67	9.0	82.00	10.0	84.67	10.5	84.67	11.1	85.00	10.6
Gisette	–	–	91.43	9.0	93.27	11.2	93.85	14.5	93.87	13.8	94.33	15.6	94.02	15.4
Geom. Mean	–	–	*79.25*	*3.5*	*78.59*	*3.9*	*79.93*	*4.5*	*81.53*	*4.7*	*80.71*	*4.8*	*80.53*	*4.8*
Arith. Mean	–	–	*80.51*	*4.2*	*80.10*	*4.8*	*80.95*	*5.6*	*82.25*	*5.8*	*81.74*	*6.1*	*81.44*	*6.1*

Table B.12: MLP classifier, IWSS$_r^2$ and CMIM-based re-ranking with block sizes B.

DataSet	**IWSS$_r^2$**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	87.10	2.9	88.71	2.7	87.10	2.8	87.10	2.8	87.10	2.8	87.10	2.8	87.10	2.9
Leukemia	91.67	2.9	91.67	2.7	91.67	2.8	91.67	2.9	91.67	2.9	91.67	2.9	91.67	2.9
Lymphoma	85.42	8.0	69.79	5.5	76.04	6.6	78.13	7.1	83.33	7.8	85.42	7.5	85.42	7.7
DLBCL	85.11	2.7	85.11	2.5	82.98	2.6	82.98	2.7	82.98	2.7	85.11	2.7	85.11	2.7
Prostate	88.24	4.8	89.71	4.1	91.18	4.6	91.91	4.5	91.91	4.7	91.18	4.7	88.24	4.6
LungCancer	95.58	2.3	96.13	2.3	95.58	2.3	95.58	2.3	95.58	2.3	95.58	2.3	95.58	2.3
GCM	52.63	20.9	44.74	7.6	49.47	9.4	46.84	12.5	53.68	13.4	54.74	14.9	52.63	14.8
Arcene	87.00	5.8	84.00	3.6	86.00	4.8	84.00	5.3	85.00	5.6	87.00	5.7	87.00	6.0
Madelon	63.80	4.5	63.95	4.4	63.85	4.5	63.75	4.5	63.75	4.5	63.75	4.5	63.75	4.5
Dorothea	91.38	11.6	92.75	3.8	93.25	5.6	93.13	7.2	93.00	7.8	93.00	8.1	92.50	8.3
Dexter	82.33	10.5	77.00	4.2	78.67	8.0	82.00	10.8	82.33	10.5	82.33	10.5	82.33	10.5
Gisette	–	–	89.53	7.2	92.62	16.0	94.18	21.0	94.48	28.3	95.22	34.0	–	–
Geom. Mean	*81.65*	*5.48*	*79.53*	*3.9*	*81.11*	*4.9*	*81.19*	*5.5*	*82.68*	*5.8*	*83.32*	*5.9*	*81.74*	*5.1*
Arith. Mean	*82.75*	*6.99*	*81.09*	*4.2*	*82.37*	*5.8*	*82.60*	*7.0*	*83.73*	*7.8*	*84.34*	*8.4*	*82.85*	*6.1*

Table B.13: TAN classifier, IWSS$_r^2$ and CMIM-based re-ranking with block sizes B.

DataSet	**IWSS^2_r**		**B=5**		**B=10**		**B=20**		**B=30**		**B=40**		**B=50**	
	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts	Acc	#atts
Colon	87.10	2.9	85.48	2.6	87.10	2.9	87.10	2.9	87.10	2.9	87.10	2.9	87.10	2.9
Leukemia	88.89	1.9	88.89	1.8	88.89	1.8	88.89	1.8	88.89	1.9	88.89	1.9	88.89	1.9
Lymphoma	90.63	6.4	84.38	6.0	89.58	6.3	90.63	6.6	90.63	6.6	88.54	6.6	88.54	6.6
DLBCL	82.98	1.7	82.98	1.3	82.98	1.3	82.98	1.3	82.98	1.4	82.98	1.6	82.98	1.7
Prostate	91.18	4.1	90.44	4.4	88.24	4.1	86.76	4.1	89.71	4.0	89.71	4.0	89.71	4.1
LungCancer	96.13	1.9	96.13	1.5	96.13	1.9	96.13	1.9	96.13	1.9	96.13	1.9	96.13	1.9
GCM	66.84	19.0	61.05	8.2	64.21	11.3	64.74	12.9	65.79	14.0	64.74	14.3	60.53	14.2
Arcene	95.00	5.3	93.00	4.2	91.00	4.9	94.00	5.0	95.00	5.1	92.00	5.3	92.00	5.3
Madelon	67.15	3.6	67.15	3.8	67.15	3.6	67.15	3.6	67.15	3.6	67.15	3.6	67.15	3.6
Dorothea	92.50	8.3	93.38	3.6	93.50	4.5	93.38	5.0	92.63	5.6	92.88	5.7	92.75	5.7
Dexter	86.67	12.9	79.00	6.8	85.67	10.9	86.67	12.8	86.33	13.1	87.33	13.1	86.33	13.1
Gisette	94.67	46.2	89.78	8.0	92.37	16.0	92.60	16.6	93.73	27.3	93.70	23.7	94.35	33.5
Geom. Mean	*86.06*	*5.5*	*83.62*	*3.7*	*84.97*	*4.4*	*85.31*	*4.6*	*85.74*	*4.9*	*85.33*	*4.9*	*84.81*	*5.1*
Arith. Mean	*86.64*	*9.5*	*84.31*	*4.4*	*85.57*	*5.8*	*85.92*	*6.2*	*86.34*	*7.3*	*85.93*	*7.1*	*85.54*	*7.9*

Table B.14: AODE classifier, IWSS^2_r and CMIM-based re-ranking with block sizes B.

B. CMIM RE-RANKING USING DIFFERENT CLASSIFIERS FOR IWSS AND IWSS WITH REPLACEMENT

Bibliography

[1] N. Abe. An iterative method for multi-class cost-sensitive learning. In *In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 3–11, 2004. 111

[2] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26, New York, NY, USA, 2006. ACM Press. 132, 145, 147, 153

[3] L. H. Armitage and P. G. Enser. Analysis of user need in image archives. *Journal of Information Science*, 23(4):287–299, 1997. 132

[4] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999. 9, 129, 131

[5] C. Barry and L. Schamber. Users' Criteria for Relevance Evaluation: A Cross-situational Comparison. *Information Processing and Management*, 34(2-3):219–236, 1998. 162

[6] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5:537–550, 1994. 26, 28, 29, 93

[7] R. Bekkerman, K. Eguchi, and J. Allan. Unsupervised non-topical classification of documents. Technical report, Center of Intelligent Information Retrieval, Massachusetts Univ., 2006. 117

[8] R. Bekkerman, A. McCallum, and G. Huang. Automatic categorization of email into folders: Bechmark experiments on Enron and SRI corpora. Technical report, Department of Computer Science. University of Massachusetts, Amherst., 2005. 9, 11, 116, 117, 123, 138

[9] A. Berson and K. Thearling. *Building Data Mining Applications for CRM*. McGraw-Hill, Inc., New York, NY, USA, 1999. 16

[10] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992. 24, 120

[11] R. Caruana and A. Niculescu-Mizil. Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–78. ACM, 2004. 17

[12] P. K. Chan and S. J. Stolfo. Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 164–168. AAAI Press, 1998. 110

[13] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at www.csie.ntu.edu.tw/∼cjlin/libsvm. 105, 120

[14] S.-F. Chang, J. R. Smith, M. Beigi, and A. Benitez. Visual information retrieval from large distributed online repositories. *Commun. ACM*, 40(12):63–71, 1997. 132

[15] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín. Searching in metric spaces. In *ACM Computing Surveys*, pages 273–321, 2001. 24

[16] N. V. Chawla, K. W. Bowyer, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002. 3, 110, 111, 119

[17] N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explorations Newsletter*, 6(1):1–6, 2004. 110, 111

[18] C. Cool and A. Spink. Issues of context in information retrieval (IR): an introduction to the special issue. *Inormation Processing Management*, 38(5):605–611, 2002. 162

[19] S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993. 23

[20] F. Crestani and I. Ruthven, editors. *Information Context: Nature, Impact, and Role*, volume 3507 of *Lecture Notes in Computer Science*. Springer, 2005. 162

[21] F. Crestani and I. Ruthven. Introduction to special issue on contextual information retrieval systems. *Information Retrieval*, 10(2):111–113, April 2007. 162

[22] B. Dasarathy. Nearest neighbor (NN) norms: NN pattern recognition classification techniques. *IEEE Computer Society Press*, 1991. 23

[23] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006. 10, 39, 41, 59, 82, 101, 102, 121, 150

[24] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification and scene analysis*. Wiley New York, 1973. 18

[25] Y. El-Manzalawy and V. Honavar. *WLSVM: Integrating LibSVM into Weka Environment*, 2005. Software available at www.cs.iastate.edu/∼yasser/wlsvm. 120

[26] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, June 2006. 13

[27] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1993. 3

[28] T. A. Feo and M. G. Resende. Greedy randomized adaptive search procedures. *Global Optimization*, 6(2):109–133, March 1995. 72

[29] A. Finn, N. Kushmerick, and B. Smyth. Genre classification and domain transfer for information filtering. In *Proceedings of ECIR-02, 24th European Colloquium on Information Retrieval Research*, 2002. 9

[30] E. Fix and J. L. Hodges. Discriminatory analysis, nonparametric discrimination. Technical report, USAF scholl of Aviation Medicine, Randof field, Project 21-49-004, Rept 4, 1951. 120

[31] F. Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, 5:1531–1555, 2004. 26, 30, 93

[32] J. Flores and J. Gámez. Breeding value classification in manchego sheep: a study of attribute selection and construction. *Lecture Notes in Computer Science*, 3682:1338–1346, 2005. 27, 36

[33] J. Flores, J. A. Gámez, and J. L. Mateo. Mining the ESROM: A study of breeding value classification in manchego sheep by means of attribute selection and construction. *Computers and Electronics in Agriculture*, 60(2):167–177, 2008. 67, 79

[34] G. Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, March 2003. 27, 140

[35] S. Fox, K. Karnawat, M. Mydland, S. Dumais, and T. White. Evaluating implicit measures to improve web search. *ACM Transactions on Information Systems*, 23(2):147–168, 2005. 1059982. 163

[36] L. Freund, E. G. Toms, and C. L. A. Clarke. Modeling task-genre relationships for IR in the workspace. In *Proceedings of the 28th SIGIR Conference*, pages 441–448, Salvador, Brazil, 2005. ACM. 164

[37] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701, 1937. 41, 59, 82, 102

[38] N. Friedman and M. Goldszmidt. Building classifiers using bayesian networks. In *AAAI/IAAI, Vol. 2*, 1996. 19, 20

[39] S. Garcia and F. Herrera. An extension on ”statistical comparisons of classifiers over multiple data sets” for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008. 82, 104

[40] M. Gutlein, E. Frank, M. Hall, and A. Karwath. Large-scale attribute selection using wrappers. In *Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 332–339, 2009. 30, 74, 79, 90

[41] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003. 25, 29, 54

[42] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70, 1979. 43, 59, 82, 102

[43] F. Hopfgartner, D. Vallet, M. Halvey, and J. M. Jose. Search trails using user feedback to improve video search. In *MM'08 - Proceedings of the ACM International Conference on Multimedia, Vancouver, Canada*, pages 339–348. ACM Press, 10 2008. 132

[44] P. Ingwersen and N. Belkin. Information retrieval in context - IRiX: workshop at SIGIR 2004. *SIGIR Forum*, 38(2):50–52, 2004. 162

[45] P. Ingwersen and K. Järvelin. Information retrieval in context: IRiX. *SIGIR Forum*, 39(2):31–39, 2005. 133, 162

[46] P. Ingwersen and K. Järvelin. *The Turn: Integration of Information Seeking and Retrieval in Context*. Springer, 2006. xvi, 163

[47] I. Inza, P. Larrañaga, R. Etxeberria, and B. Sierra. Feature subset selection by bayesian network-based optimization. *Artifial Intelligence*, 123(1-2):157–184, 2000. 29

[48] A. Jaimes, M. Christel, S. Gilles, R. Sarukkai, and W.-Y. Ma. Multimedia information retrieval: what is it, and why isn't anyone using it? In *MIR '05: Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, pages 3–8. ACM, 2005. 130

[49] N. Japkowicz. The class imbalance problem: Significance and strategies. In *In Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI)*, pages 111–117, 2000. 110, 111, 113

[50] N. Japkowicz. Concept-learning in the presence of between-class and within-class imbalances. In *AI '01: Proceedings of the 14th Biennial Conference of*

the Canadian Society on Computational Studies of Intelligence, pages 67–77, London, UK, 2001. Springer-Verlag. 110

[51] N. Japkowicz and S. Shaju. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):429–449, 2002. 124

[52] F. Jensen and T. Nielsen. *Bayesian networks and decision graphs (2ed)*. Springer Verlag, 2007. 46

[53] T. Jo and N. Japkowicz. Class imbalances versus small disjuncts. *SIGKDD Explorations Newsletter*, 6(1):40–49, 2004. 110

[54] T. Joachims. Text categorization with support vector machines: learning with many relevant features. Technical Report LS-8 Report 23, University of Dortmund, 1998. 14

[55] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986. 3, 25

[56] D. Kelly and N. J. Belkin. Display time as implicit feedback: understanding task effects. In *Proceedings of the 27th SIGIR Conference*, pages 377–384, Sheffield, UK, 2004. ACM Press. 163

[57] S.-B. Kim, K.-S. Han, H.-C. Rim, and S. H. Myaeng. Some effective techniques for naive bayes text classification. *IEEE Transactions on Knowledge and Data Engineering*, 18(11):1457–1466, 2006. 115

[58] S.-B. Kim, H.-C. Seo, and H.-C. Rim. Poisson naive bayes for text classification with feature weighting. In *Proceedings of the sixth international workshop on Information retrieval with Asian languages*, pages 33–40, 2003. 114

[59] K. Kira and L. A. Rendell. A practical approach to feature selection. In *Proceedings of the ninth international workshop on Machine learning*, pages 249–256. Morgan Kaufmann Publishers Inc., 1992. 26

[60] J. Kittler. Feature set search algorithms. *Pattern Recognition and Signal Processing*, pages 41–60, 1978. 29, 78

[61] B. Klimt and Y. Yang. The ENRON corpus: a new dataset for email classification research. In *15th European Conference on Machine Learning*, pages 217–226, 2004. 117

[62] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artifical Intelligence*, 97(1-2):273–324, 1997. 29

[63] M. Kubat and S. Matwin. Addressing the curse of imbalanced training sets: one-sided selection. In *In Proceedings of the Fourteenth International Conference on Machine Learning*, pages 179–186. Morgan Kaufmann, 1997. 110

[64] D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 4–15, 1998. 115

[65] Y. Lin, Y. Lin, Y. Lee, Y. Lee, G. Wahba, and G. Wahba. Support vector machines for classification in nonstandard situations. In *Machine Learning*, pages 191–202, 2002. 110

[66] R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data, Second Edition*. Wiley-Interscience, Hoboken, USA, 2 edition, 2002. 3

[67] H. Liu and H. Motoda. *Feature Extraction Construction and Selection: a data mining perspective*. Kluwer Academic Publishers, 1998. 3, 25

[68] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, 1998. 29

[69] H. Liu, J. Sun, L. Liu, and H. Zhang. Feature selection with dynamic mutual information. *Pattern Recognition*, 42(7):1330–1339, 2009. 26

[70] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):491–502, 2005. 26, 29

[71] Y. Liu, H. T. Loh, and A. Sun. Imbalanced text classification: A term weighting approach. *Expert Systems with Applications*, 36(1):690–701, 2009. 110

[72] L. Malazizi, D. Neagu, and Q. Chaudhry. Improving imbalanced multidimensional dataset learner performance with artificial data generation: Density-based class-boost algorithm. In P. Perner, editor, *ICDM '08: Proceedings of the 8th industrial conference on Advances in Data Mining*, volume 5077 of *LNCS*, pages 165–176. Springer, 2008. 111

[73] S. Mallat and W. L. Hwang. Singularity detection and processing with wavelets. *IEEE Transactions on Information Theory*, 38:617–643, 1992. 134

[74] B. S. Manjunath, P. Salembier, and T. Sikora, editors. *Introduction to MPEG-7: Multimedia Content Description Language*. Wiley, April 2002. 134

[75] C. D. Manning, P. Raghavan, and H. Schütze. *An Introduction to Information Retrieval*. Cambridge University Press, 2008. 9, 130

[76] A. Masegosa. *Models of Supervised Classification. Applications to Genomics and Information Retrieval*. PhD thesis, University of Granada, 2009. 9

[77] C. Matheus and L. Rendell. Constructive induction on decision trees. In *Proceedings of the International Joint conference on Artificial Intelligence (IJCAI)*, pages 645–650. Morgan And Kaufmann, 1989. 3, 25

[78] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI/ICML-98 Workshop on Learning for Text Categorization*, pages 41–48, 1998. 120, 148, 153

[79] T. M. Mitchell. *Machine learning*. McGraw Hill, 1996. 10

[80] S. Nakariyakul and D. P. Casasent. An improvement on floating search algorithms for feature subset selection. *Pattern Recognition*, 42:1932–1940, 2009. 29, 68

[81] N. O'Connor. Summer school on multimedia semantics: Image and video processing, 2007. 135

[82] M. Pazzani. Searching for dependencies in bayesian classifiers. In: Learning from Data: AI and Statistics V. Springer-Verlag, 1996. 19, 20, 21

[83] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988. 46

[84] 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, August 2005. 28, 30, 93

[85] A. Pérez, P. Larrañaga, and I. Inza. Supervised classification with conditional gaussian networks: Increasing the structure complexity from naive bayes. *International Journal of Approximage Reasoning*, 43:1–25, 2006. 20

[86] M. F. Porter. An algorithm for suffix stripping. pages 313–316, 1997. 147

[87] R. C. Prati, G. Bastista, and M. C. Monard. Class imbalances versus class overlapping: an analysis of a learning system behavior. In *Third Mexican international conference on artificial intelligence (MICAI)*, pages 312–321. LNAI, 2004. 110

[88] W. K. Pratt. *Digital Image Processing*. Wiley-Interscience, 4 edition, February 2007. 134

[89] P. Pudil, J. Novovicová, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15(10):1119–1125, 1994. 29, 73

[90] J. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986. 22

[91] B. Raskutti and A. Kowalczyk. Extreme re-balancing for SVMs: a case study. *SIGKDD Explorations Newsletter*, 6(1):60–69, 2004. 110

[92] C. V. Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 1979. 15

[93] T. Rose, M. Stevenson, and M. Whitehead. The Reuters corpus volume 1- from yesterday's news to tomorrow's language resources. In *In Proceedings of the Third International Conference on Language Resources and Evaluation*, 2002. 9

[94] R. Y. Rubinstein and B. Melamed. *Modern Simulation and Modeling (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 1998. 114

[95] S. Rudinac, G. Zajic, M. Uscumlic, M. Rudinac, and B. Reljin. Comparison of CBIR systems with different number of feature vector components. In *SMAP '07: Proceedings of the Second International Workshop on Semantic Media Adaptation and Personalization*, pages 199–204, Washington, DC, USA, 2007. IEEE Computer Society. 140

[96] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: a power tool for interactive contentbased image retrieval. *Circuits and Systems for Video Technology, IEEE Transactions on*, 8(5):644–655, 1998. 132

[97] R. Ruiz, J. S. Aguilar, and J. Riquelme. Best agglomerative ranked subset for feature selection. In *JMLR: Workshop and Conference Proceedings vol. 4 (New Challenges for feature selection)*, pages 148–162, 2009. 31, 78, 79, 90

[98] R. Ruiz, J. C. Riquelme, and J. S. Aguilar-Ruiz. Incremental wrapper-based gene selection from microarray data for cancer classification. *Pattern Recognition Letters*, 39:2383–2392, 2006. 27, 31, 36, 37, 38, 40, 45, 53, 67, 79

[99] I. Ruthven, P. Borlund, P. Ingwersen, N. Belkin, A. Tombros, and P. Vakkari, editors. *Proceedings of the 1st IIiX Symposium*, Copenhagen, Denmark, 2006. 162

[100] M. Sahami. Learning limited dependence Bayesian classifiers. In *Second International Conference on Knowledge Discovery in Databases*, 1996. 20

[101] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. Technical report, Cornell University, 1987. 117

[102] J. S. Sánchez, F. Pla, and F. J. Ferri. Prototype selection for the nearest neighbour rule through proximity graphs. *Pattern Recognition Letters*, 18:507–513, 1997. 24

[103] S. Scott and S. Matwin. Feature engineering for text classification. In *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 379–388, 1999. 117

[104] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34:1–47, 2002. 9, 10

[105] H. T. Shen, B. C. Ooi, and X. Zhou. Towards effective indexing for very large video sequence database. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 730–741, Baltimore, Maryland, USA, 2005. ACM. 135

[106] T. Sikora. The MPEG-7 visual standard for content description-an overview. 11(6):696–702, June 2001. 134

[107] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and TRECVid. In *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, pages 321–330, New York, NY, USA, 2006. ACM Press. 131, 149, 154, 165

[108] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1349–1380, 2000. 131, 132, 133

[109] P. Somol, P. Pudil, and J. Kittler. Fast branch & bound algorithms for optimal feature selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7):900–912, 2004. 29

[110] E. Stamatatos. Author identification: Using text sampling to handle the class imbalance problem. *Information Processing & Management*, 44(2):790–799, 2008. 111, 117

[111] M. Swain and D. Ballard. Indexing via color histograms. In *Proceedings of the Third International Conference on Computer Vision*, pages 390–393. IEEE Computer Society, 1990. 134

[112] A. Tombros, I. Ruthven, and J. M. Jose. How users assess web pages for information seeking. *Journal of the American Society for Information Science and Technology*, 56(4):327–344, 2005. 162, 164

[113] T. Urruty, C. Djeraba, and J. M. Jose. An efficient indexing structure for multimedia data. In *MIR '08: Proceeding of the 1st ACM international conference on Multimedia information retrieval*, pages 313–320. ACM, 2008. 135

[114] R. Villa, N. Gildea, and J. M. Jose. Facetbrowser: a user interface for complex search tasks. In *MM '08: Proceeding of the 16th ACM international conference on Multimedia*, pages 489–498, New York, NY, USA, 2008. ACM. 132, 146

[115] R. Villa, N. Gildea, and J. M. Jose. A study of awareness in multimedia search. In *JCDL '08: Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, pages 221–230, New York, NY, USA, 2008. ACM. 146, 154

[116] G. Webb, J. Boughton, and Z. Wang. Not so naive bayes: Aggregating one-dependence estimators. *Machine Learning*, 58(1):5–24, 2005. 137

[117] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of 24rd International Conference on Very Large Data Bases*, pages 194–205, New York city, New York, USA, 1998. Morgan Kaufmann. 135

[118] G. M. Weiss. Mining with rarity: a unifying framework. *SIGKDD Explorations Newsletter*, 6(1):7–19, 2004. 110, 153

[119] A. P. White and W. Z. Liu. Technical note: Bias in information-based measures in decision tree induction. *Maching Learning*, 15(3):321–329, 1994. 140

[120] R. W. White, I. Ruthven, and J. M. Jose. A study of factors affecting the utility of implicit relevance feedback. In *Proceedings of the 28th SIGIR Conference*, pages 35–42, Salvador, Brazil, 2005. ACM. 163

[121] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945. 39, 101, 121, 150, 154

[122] I. H. Witten, Z. Bray, M. Mahoui, and W. J. Teahan. Text mining: A new frontier for lossless compression. In *Proceedings of Data Compression Conference*, pages 198–207, 1999. 117

[123] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*. Morgan Kaufmann, 2005. 40, 54, 79, 105, 117

[124] L. Yan, M. Fassino, and P. Baldasare. Enhancing the lift under budget constraints: an application in the mutual fund industry. In R. Grossman, R. Bayardo, and K. P. Bennett, editors, *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 509–515. ACM, 2005. 16

[125] R. Yan and A. G. Hauptmann. Co-retrieval: A boosted reranking approach for video retrieval. In *Conference on Image and Video Retrieval (CIVR-04)*, pages 60–69, 2004. 134, 148

[126] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems*, 13(2):44–49, 1998. 29

[127] L. Yu and H. Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. *In Proceedings of The Twentieth International Conference on Machine Leaning (ICML-03)*. 29

[128] L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, 5:1205–1224, 2004. 26, 54

[129] S. C. Yusta. Different metaheuristic strategies to solve the feature selection problem. *Pattern Recognition Letters*, 30(5):525–534, 2009. 73

[130] Z. Zheng, X. Wu, and R. K. Srihari. Feature selection for text categorization on imbalanced data. *SIGKDD Explorations Newsletter*, 6(1):80–89, 2004. 110

[131] Z.-H. Zhou and X.-Y. Liu. On multi-class cost-sensitive learning. In *AAAI*, 2006. 111