MINISTERUL EDUCAȚIEI, CERCETĂRII, TINERETULUI ȘI SPORTULUI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTY OF COMPUTER SCIENCE AND AUTOMATION

Eng. Camelia Lemnaru (Vidrighin Bratu)

PhD THESIS

STRATEGIES FOR DEALING WITH REAL WORLD CLASSIFICATION PROBLEMS

Scientific Advisor:
Prof.Dr.Eng.Sergiu NEDEVSCHI

Committee:

PRESIDENT: Prof.dr.ing. ***Liviu Miclea*** – Dean, Faculty of Computer Science and Automation, Technical University of Cluj-Napoca;

MEMBERS: Prof.dr.ing. ***Sergiu Nedevschi*** – Supervisor, Technical University of Cluj-Napoca;
Prof.dr.ing. ***Mircea Petrescu*** – Reviewer, University "Politechnica" of Bucharest;
Prof.dr.ing. ***Vladimir Cretu*** – Reviewer, University "Politechnica" of Timisoara;
Prof.dr.ing. ***Rodica Potolea*** – Reviewer, Technical University of Cluj-Napoca.

str. Memorandumului nr.28, 400114 Cluj-Napoca, România
tel. +40-264-401200, fax +40-264-592055, secretariat tel. +40+264-202209, fax +40-264-202280
http://www.utcluj.ro/

The beginning of knowledge is the discovery of something we do not understand.

Frank Herbert

~ To my family ~

ACKNOWLEDGEMENTS

I would like to express my gratitude towards my scientific advisor, Prof. dr. eng. Sergiu Nedevschi, for all the guidance and constructive observations he has given me throughout my PhD research period.

Also, this thesis would probably not have been completed without the constant involvement and invaluable support provided by Prof. dr. eng. Rodica Potolea, to whom I will be forever grateful. Thank you for everything you have taught me during this period.

I would also like to thank all the undergraduate and master students who have contributed to my research, for their devotion and constant involvement.

Last, but not least, I dedicate this thesis to my family, for their constant encouragement and for being who they are.

Table of Contents

Index of Figures

Index of Tables

List of Abbreviations

AB = *AdaBoost.M1 ensemble classifier*
Acc = *accuracy, performance metric*
AUC = *Area Under the ROC Curve, performance metric*
BAcc = *balanced accuracy, performance metric*
C = *complexity of a classification model*
C4.5 = *C4.5 decision tree learner*
CFS = *Correlation-based Feature Selection`*
CS = *Cost-Sensitive strategy*
ECSB = *Evolutionary Cost-Sensitive Balancing*
EM = *Expectation Maximization*
EUS = *Evolutionary Under-Sampling*
FM = *F-measure*
GM = *geometric mean, performance metric*
IAR = *Instances per Attribute Ratio*
IBk = *k instance-based (k-nearest neighbor) classifier, WEKA implementation*
ICET = *Inexpensive Classification with Expensive Tests*
IR = *Imbalance Ratio*
J4.8 = *WEKA implementation of the C4.5 decision tree classifier*
kNN = *k-nearest neighbor classifier*
MC = *Metacost cost-sensitive classifier*
MLP = *Multilayer Perceptron classifier*
NB = *Naïve Bayes classifier*
ProICET = *enhanced version of the ICET method*
ROC = *Receiver-Operating Characteristic*
SVM = *Support Vector Machines classifier*
SVMEns = *SVM Ensemble*
TN_{rate} = *True Negative rate*
TP_{rate} = *True Positive rate*
UCI = *University California Irvine*
WEKA = *Waikato Environment for Knowledge Analysis*

Abstract

A series of challenges have recently emerged in the data mining field, triggered by the rapid shift in status from academic to applied science and the resulting needs of real-life applications. The current thesis is concerned with classification tasks and related issues which may appear in real-world scenarios, such as: incomplete records and irrelevant and/or redundant pieces of information, imbalanced class distribution and imbalanced error costs. Also, there is no universally best classifier which performs better than all the others on every possible problem, given any evaluation metric. Moreover, no general rules which indicate the appropriate metric to select in a certain context exist. Translating the data characteristics and problem goals into appropriate performance metrics, selecting the most appropriate classifier with the best parameter settings are therefore essential points in achieving a successful data mining process. Moreover, application domains may impose specific constraints on the data mining process, such as having an interpretable classification model, or a reasonable training time, or the capacity to perform classification on a large number of classes, each having a limited amount of training instances.

The current thesis ascertains the problem statement and provides an analysis of existing approaches for the major theoretical problems tackled – and, in some cases, also systematic empirical studies. Also, it proposes a series of novel methods for improving the behavior of traditional classifiers in such imperfect scenarios. In the data pre-processing step, the current thesis introduces an original global imputation method, based on non-missing data and a novel joint pre-processing methodology, which proposes an information exchange between data imputation and feature selection. Also, an original subset combination method for improving the stability of feature selection across different problems and providing an assessment of the baseline performance of feature selection in a new problem is presented.

For the actual processing step, an original meta-classification strategy has been proposed – Evolutionary Cost-Sensitive Balancing. The method performs a genetic search in order to simultaneously tune the base classifier's parameters and identify the most appropriate cost matrix, which is then employed by a cost-sensitive meta-learner, in conjunction with the base classifier. An advantage of the method, besides its generality, is the fact that the cost matrix is determined indirectly. Also, this thesis proposes a series of significant enhancements to an existing cost-sensitive classifier: the employment of a single population technique and elitism, with rank-based fitness assignment for parent selection, and increasing search variability via appropriate recombination operators.

An original meta-learning framework for automated classifier selection is presented in the case studies section, together with a method for baseline performance assessment. Also in the case studies chapter, a series of additional constraints imposed by specific domains are analyzed, and original solutions are provided, in which the general flow of the data mining process may be altered to accommodate for the specific domain needs, without restricting the solutions to the specific domains they have been initially designed for.

Keywords: *data mining, classification, incomplete data, imputation, attribute/feature selection, joint pre-processing methodology, imbalanced class distribution, cost-sensitive learning, evaluation metrics, automated classifier selection, classification case studies.*

1 Introduction

1.1 Motivation

A series of challenges have recently emerged in the data mining field, triggered by the rapid shift in status from academic to applied science and the resulting needs of real-life applications. The current thesis is concerned with classification tasks. Therefore, the general context is that of labeling (a potentially large volume of) data which may contain incomplete records and irrelevant and/or redundant pieces of information. Interesting cases tend to occur less frequently; therefore they possess poorer representation in the available data. However, the correct identification of such cases is often of utmost importance. In the following, the motivation behind the main objectives of the thesis is presented.

A first issue tackled is that of incomplete data, i.e. the existence of unknown values in the available data. While a common occurrence in real-world problems, it represents a challenge for classification, since most algorithms have not been originally designed to deal with missing data, and most of them employ simple and rather inefficient adaptations do deal with this issue. Several more evolved missing data techniques (MDTs) are available in literature, either as pre-processing methods – filter or imputation-based – or as embedded approaches. Filter methods attempt to eliminate missing records/attributes such that subsequent analyses are performed on complete data. Imputation methods try to reach the same goal, but by searching for a replacement value for each missing one. Embedded approaches designate mechanisms for dealing with missing data specific to a certain learning algorithm, such as the strategies employed by some decision trees. The problem with pre-processing MDTs is that the majority have been developed to alleviate the effect of incomplete data on subsequent statistical analyses, and not on improving a classification task – for which different assumptions may be necessary. Also, filters tend to introduce bias in the learning process and reduce the volume of available data, while existing imputation methods either require strong assumptions which don't generally hold in real situations (e.g. multiple imputation), or employ knowledge about the data distribution (e.g. expectation maximization), which is not always accessible. Computational complexity may also represent an issue for some imputation techniques. Embedded methods are restricted to the particular learner (or category of learners). Thus, an imputation method which is focused on improving subsequent classification performance and does not require strong assumptions is of interest. Also, though imputation methods make assumptions related to the existence of correlations between the attributes, they do not perform an explicit analysis of the strength of the correlations, and the class attribute is generally discarded. Selecting – for each attribute to be imputed – a subset of strongly predictive attributes (in which the class may be included) could produce a better imputation quality with respect to the subsequent classification step.

Another important aspect related to the preparation of the available data for the actual classification step is associated with dimensionality reduction. Such an operation may produce several benefits in the overall economy of the data mining process, such as: performance improvement, by alleviating the curse of dimensionality and improving the model generalization capabilities, speed-up by reducing the computational effort, improved model interpretability by reducing its complexity, cost reduction by avoiding "expensive" features. The feature selection problem has been studied both intensively and extensively in the literature, with available solutions being split into: filter, wrapper and embedded approaches. For the purpose of performance improvement, wrappers generally provide the most appropriate strategy. However, the best wrapper instantiation depends on the problem particularities and the specific learner used for classification. One particular wrapper method may produce significant improvements in one given problem for a specific algorithm, while fail to improve the classification performance of the same algorithm on a different problem

(or a different optimization objective), or the same problem and a different learning algorithm. Also, it is not clear how wrapper feature selection affects the initial choice of the learning algorithm, and which strategy would be appropriate if the ranking of classifiers is affected: prefer a stable, yet not the best performance over a better, less stable one?

In many real-world applications, instances belonging to different classes may be non-uniformly distributed in the available data, meaning that one class may contain fewer instances in comparison with the others. However, the correct identification of underrepresented classes is generally of increased importance, which results in different errors possessing different degrees of gravity. Thus, between the different classes involved in the recognition process, imbalance may occur at two levels: one at data level, which results in an imbalanced class distribution problem, and one at the level of the impact produced by different errors, which results in imbalanced error costs problem. Each of the two resulting issues is addressed in this thesis.

There are few systematic studies in literature which analyze how an imbalanced class distribution affects learning algorithms, using a variety of learning strategies, problems and under several evaluation metrics. Techniques which attempt to address class imbalance are available in the literature, either as sampling methods, algorithm-based methods (including modifications to existing algorithms or newly proposed methods, built to deal with the imbalance intrinsically) and hybrid methods (which include strategies which mix data- with algorithm-oriented techniques, or apply meta-approaches, such as cost-sensitive learning). While each category provides prominent representative methods, the following drawbacks appear: to maximize their effect, sampling methods need to be matched to the specific learning method employed; several sampling methods require the analyst to set the amount of re-sampling needed – which requires both experience and intuition or a considerable amount of time; the application of algorithm-based methods is restricted to the specific categories of classifiers they have been designed for; cost-sensitive hybrid strategies require the analyst to set the cost-matrix, a difficult task – similar to setting the amount of re-sampling – but which could result, in some areas, into serious social dilemmas, such as putting a price tag on human life. Therefore, a general method aimed at alleviating the effect of the class imbalance problem and is applicable to any existing classification method without requiring extensive familiarity with the mechanisms involved or detailed domain knowledge, is of interest.

While the imbalanced class distribution problem is theoretically independent of the application domain, imbalanced error costs appear in areas in which the impact produced by different errors varies according to the error particularity. A straightforward example is the medical diagnosis problem, in which failing to identify a positive diagnosis is almost unacceptable, whereas a certain level of false positive predictions is considered manageable. In such domains, the effort of acquiring data values has to be considered also: medical tests are usually costly (economically) and may produce patient discomfort; in addition, collecting test data is time consuming. Each of these aspects has an implication on whether it is practical to perform a certain test or not. Cost-sensitive learning addresses these particularities, using two important types of costs to model the above-mentioned asymmetries: misclassification and test costs. Most cost-sensitive algorithms existing in literature focus on a single type of cost: either misclassification or test costs. Significantly less work has been invested towards developing a technique which considers both, the most prominent such method being the ICET algorithm. While the strategy proposed by ICET is promising, the initial work on ICET lacked a comprehensive study of the misclassification cost component. Also, an attempt to improve the algorithm behavior via a more appropriate selection of the genetic strategies employed might be of interest.

Although the central objective of any classification algorithm is the minimization/ maximization of some objective function on a data sample, different classification algorithms

employ different theories to accomplish this goal. The main consequence of this fact is formulated by the *no-free lunch theorem*, which essentially states that there is no universally best classification method. This triggers the need to select the appropriate learning scheme for a given problem. In addition, the classification algorithms and/or their resulting models posses several particularities, which can be evaluated when determining their suitability for a specific context: robustness to noise, scalability, speed (prediction speed mostly, but – in certain scenarios – training speed also), model interpretability, robustness to irrelevant/redundant features, robustness to numeric features, etc. Such aspects have been considered throughout the thesis, according to specific problem particularities and goals.

The initial assessment of the appropriate learning scheme for the analyzed data represents a complex and time-consuming task even for an experienced data analyst. Consequently, several systems which employ meta-learning strategies in the attempt to provide a certain degree of automation to the classifier selection problem have been proposed in literature. They utilize various dataset meta-features and specific prediction strategies in the attempt to indicate the most appropriate learning scheme for a new problem. Besides their individual drawbacks (too few meta-features, speed, necessity for user involvement), a common disadvantage of such systems is that employ the accuracy alone as the classification performance metric, which has proven to be insufficient in domains which present class- or error-imbalance issues. Therefore, a scalable framework which brings together efficiently the tools necessary to analyze new problems and make predictions related to the learning algorithms' performance, while keeping the analyst's involvement at a minimum, is of interest.

In addition to the principal motivation presented above, the current thesis tackles specific application-related constraints which may be imposed on the general data mining process, such as generating an interpretable classification model, using a reasonable training time, or the capacity to perform classification on a large number of classes, each having a limited amount of training instances. Also, specific data mining process instantiations in the following domains have been explored: written signature recognition, handwritten documents transcription, network intrusion detection, community detection, opinion mining and spam filtering.

The thesis acknowledges the existence of several other issues involved in a classification data mining process, such as those related to: data visualization techniques, noise reduction, feature construction or aggregation, knowledge presentation strategies. However, it does not address those issues specifically.

1.2 Thesis Overview

Chapter 2 presents a general view on data mining as a field and the data mining process, outlining the steps involved in the process (section 2.1), the main types of data mining tasks (section 2.2), an overview of the main types of classification algorithms available in literature (section 2.3) and classifier evaluation strategies and measures (section 2.4).

Chapter 3 addresses the missing data problem, starting with the problem statement (section 3.1) and providing a structured analysis of existing missing data techniques (section 3.2). Section 3.3 proposes an original global imputation method based on non-missing attribute values. The method employs a different neural network classifier ensemble model to impute values for each incomplete attribute in turn. It makes use of the existing correlations between the attributes – including the class attribute, to learn a different imputation model for each class. The main assumption needed for the method to produce good results is the existence of a complete data kernel. The evaluation of the method using an univariate MCAR incompleteness mechanism is presented in section 3.3.2.

The first two sections of Chapter 4 (4.1 and 4.2) present an overview on the feature selection problem and the main strategies for performing feature selection in practice. Section 4.3 proposes a combination of different subset selection strategies, via individual feature voting. The purpose is to remove the bias introduced by different search methods in selecting a specific feature subset, thus improving the stability of the method across several problems. A systematic analysis on different wrapper combinations and an empirical validation of the proposed combination method are presented in section 4.3.

In Chapter 5, a methodology for joining two previously independent pre-processing steps is presented and its effect on the classification step is evaluated (section 5.2). The method proposes the information exchange between the data imputation and feature selection step – most specifically to employ only a subset of attributes with high prediction power for an attribute to build the imputation model for that attribute. Also, the imputation of strongly predictive attributes with the class is important, since weakly predictive attributes tend to be eliminated via feature selection.

Chapter 6 addresses the concept of cost in classification problems. Section 6.2 provides a systematic review of the main techniques for cost-sensitive learning – both methods which tackle test costs and those which focus on misclassification costs, with ICET being one of the most prominent approaches which considers both types of cost. Section 6.3 proposes a series of enhancements on the genetic component of the algorithm, meant to increase the search variability while always propagating the best solutions in the next generation unchanged – thus avoiding getting trapped at local optima and the loss of good candidates. Section 6.3.3 provides a comprehensive analysis on the performance of the enhanced ProICET method, with a comparative evaluation of the misclassification cost component and total cost on benchmark medical datasets, and a study on the effect of stratification as a strategy for tackling error cost asymmetries. Section 6.3.4 presents a case study for the application of ProICET to a real prostate cancer medical problem, which comes as a result of the involvement in the CEEX research grant no. 18/2005 – IntelPRO.

Chapter 7 tackles the class imbalance problem – starting with a systematic analysis of the effect of class imbalance on the performance of several types of classification methods, on a large sample of benchmark datasets, and under various performance metrics (section 7.1.3). An original dataset meta-feature is proposed – the Instances per Attributes Ratio, which aggregates size and complexity information and can be used in conjunction with the Imbalance Ratio to provide a general indication on the expected performance of classifiers in a new imbalanced problem. Section 7.2 supplies a thorough analysis of the main approaches for dealing with class imbalance from literature. Section 7.3 presents an original meta-approach for improving the behavior of base classifiers in imbalanced scenarios. The method employs a genetic search to find appropriate parameter settings and a cost matrix which are then applied to the base classifier in conjunction with a cost-sensitive learning strategy. Extensive comparative evaluations have been performed on the newly proposed method, described in section 7.3.2.

Chapter 8 presents a number of case studies, which adapt the general data mining process flow to specific requirements of several application domains, and provides a series of solutions with a higher level of generality, such that they are not restricted to the exact problem constraints they are designed for. All solutions investigate several data mining process steps, and some apply methods from previous chapters. Section 8.1 tackles the problem of automated classifier selection, with an original proposal for an automated framework to perform this task (section 8.1.2). Section 8.2 presents three original methods which employ data partitioning to evolve multiple sub-models and combine their predictions to achieve increased classification performance and/or scalability: an arbiter-combiner technique (section 8.2.1) and two hierarchical models, one based on clustering and

classification sub-models, applied in the context of offline signature recognition (section 8.2.2); a second, which consists of a multiple classifier which combines the predictions of binary classifiers, and employs an additional classifier for handling difficult to classify instances (section 8.2.3). Section 8.3 addresses problems related to speed-up and scalability, and proposes two original systems: a parallel version of the SPRINT decision tree classifier (section 8.3.2) and a lightweight parallel genetic algorithms framework, which utilizes the General-Purpose Computation on Graphics Hardware paradigm (section 8.3.3). Also, a distributed version for the original method proposed in Chapter 7 is presented (section 8.3.1). Section 8.4 explores several specific data mining applications: user type identification in adaptive e-learning systems (section 8.4.1), handwritten document transcription of historical documents (section 8.4.2), community structure detection in social mining (section 8.4.3), a semi-supervised opinion mining technique (section 8.4.4) and spam prediction in spam filtering (section 8.4.5). The user-type identification model presented in section 8.4.1 has been applied within PNII research grant no. 12080/2008 – SEArCH.

The concluding remarks, with discussions on the original elements of the thesis are presented in Chapter 9. A list of research grants, important publications and citations is available after the references section.

2 Data mining: Concepts and Definitions

The search for patterns in data is a human endeavor that is as old as it is ubiquitous, and has witnessed a dramatic transformation in strategy throughout the years. Whether we refer to hunters seeking to understand the animals' migration patterns, or farmers attempting to model harvest evolution, or turn to more current concerns, like sales trend analysis, assisted medical diagnosis, or building models of the surrounding world from scientific data, we reach the same conclusion: hidden within raw data we could find important new pieces of information and knowledge.

Traditional approaches for deriving knowledge from data rely strongly on manual analysis and interpretation. For any domain – scientific, marketing, finance, health, business, etc. – the success of a traditional analysis depends on the capabilities of one/more specialists to read into the data: scientists go through remote images of planets and asteroids to mark interest objects, such as impact craters; bank analysts go through credit applications to determine which are prone to end in defaults. Such an approach is slow, expensive and with limited results, relying strongly on experience, state of mind and specialist know-how.

Moreover, the volume of generated data is increasing dramatically, which makes traditional approaches impractical in most domains. Within the large volumes of data lay hidden strategic pieces of information for fields such as science, health or business. Besides the possibility to collect and store large volumes of data, the information era has also provided us with an increased computational power. The natural attitude is to employ this power to automate the process of discovering interesting models and patterns in the raw data. Thus, the purpose of the knowledge discovery methods is to provide solutions to one of the problems triggered by the information era: "data overload" [Fay96].

A formal definition of *data mining* (DM), also known – historically – as *data fishing, data dredging* (1960-), *knowledge discovery in databases* (1990-), or – depending on the domain, as *business intelligence, information discovery, information harvesting* or *data pattern processing* – is [Fay96]:

> **Definition:** *Knowledge Discovery in Databases (KDD) is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.*

By *data* the definition refers to a set of facts (e.g. records in a database), whereas *pattern* represents an expression which describes a subset of the data, i.e. any structured representation or higher level description of a subset of the data. The term *process* designates a complex activity, comprised of several steps, while *non-trivial* implies that some search or inference is necessary, the straightforward derivation of the patterns is not possible. The resulting models or patterns should be *valid* on new data, with a certain level of confidence. Also, we wish that the patterns be *novel* – at least for the system and, ideally, for the analyst – and *potentially useful*, i.e. bring some kind of benefit to the analyst or the task. Ultimately, they need to be interpretable, even if this requires some kind of result transformation.

An important concept is that of *interestingness*, which normally quantifies the added value of a pattern, combining validity, novelty, utility and simplicity. This can be expressed either explicitly, or implicitly, through the ranking performed by the DM system on the returned patterns. A short note should be made on the fact that, even if initially DM represented a component in the KDD process, responsible with finding the patterns in data, currently the two terms are used interchangeably, both being employed to refer to the overall discovery process, which is comprised of several steps, as presented in the next section.

2.1 Data Mining as Process

A reduced/idealist view of the DM process presents it as the development of computer programs which automatically examine raw data, in the search for models and regularities. In reality, performing data mining implies undergoing an entire process, and requires techniques from a series of domains, such as: statistics, machine learning, artificial intelligence, visualization. Essentially, the DM process is iterative and semi-automated, and may require human intervention in several key points.

Figure 2.1 presents a generic model for the DM process, with the main logical steps involved. The colored-background boxes represent steps for which original contributions will be presented throughout this dissertation.

Data filtering is responsible with the selection of relevant data for the intended analysis, according to the problem formulation. *Data cleaning* is responsible for handling missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies, such as to compensate for the learning algorithms' inability to deal with such data irregularities. *Data transformation* activities include aggregation, normalization and solving syntactic incompatibilities, such as unit conversions or data format synchronization (according to the requirements of the algorithms used in the processing steps). *Data projection* translates the input space into an alternative space, (generally) of lower dimensionality. The benefits of such an activity include processing speed-up, increased performance and/or reduced complexity for the resulting models.

During the *processing* steps, learning models/patterns are inferred, by applying the appropriate learning scheme on the pre-processed data. The processing activities are included in an iterative process, during which the most appropriate algorithm and associated parameter values are established (*model generation and tuning*). The correct choice of the learning algorithm, given the established goals and data characteristics, is essential. There are situations in which it is required to adapt existing algorithms, or to develop new methods in order to satisfy all requirements. Subsequently, the output model is built using the results from the model tuning loop, and its expected performance is assessed.

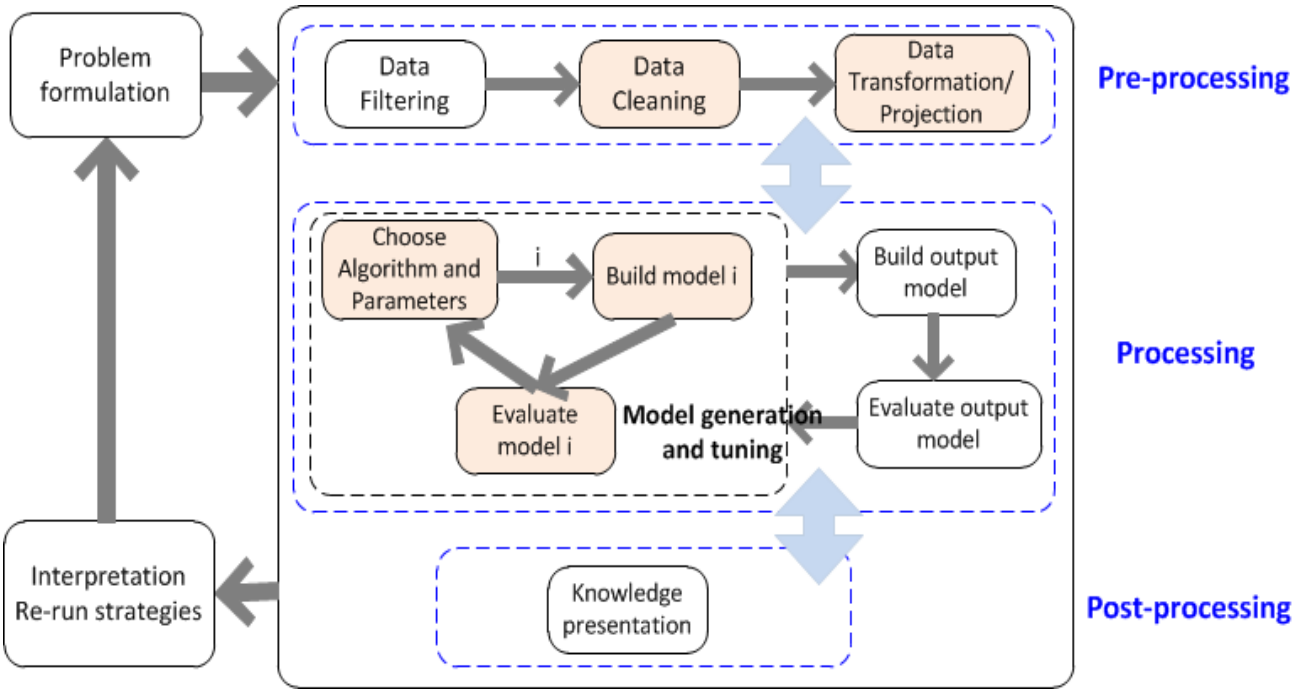


Figure 2.1 - Main steps of a data mining process

Knowledge presentation employs visualization methods to display the extracted knowledge in an intuitive, accessible and easy to understand manner. Decisions on how to proceed with future iterations are made based on the conclusions reached at this point.

DM process modeling represents an active challenge, through their diversity and uniqueness within a certain application. All process models contain activities which can be conceptually grouped, into the three types: pre-processing, processing and post-processing. Several standard process models exist in literature, the most important being: William's model, Reinartz' model, CRISP-DM, I-MIN or Redpath's model [Bha08]. Each model specifies the same process steps and data flow; they differ in the control flow. Essentially, they all try to achieve maximum possible automation.

2.2 Knowledge Extraction Tasks

The purpose of a data mining system is defined through the desired employment mode. According to [Fay96], two main categories of goals can be distinguished: *verification* and *discovery*. With verification, the system is limited to validating user hypotheses (which can be accomplished through OLAP techniques as well), but for discovery the system infers new patterns autonomously – this is actually pure DM. *Description* and *prediction* are the main types of discovery tasks.

Description tasks generate models and present them to the user in an easy-to-understand form. *Summarization* and *visualization*, *clustering*, *link analysis* and *outlier analysis* represent descriptive tasks. Prediction tasks generate models for predicting future behavior of certain entities. *Classification*, *regression* and *time series prediction* are the main tasks falling in this category.

A *dataset* is a finite set T of elements, named instances, or examples:

$$T = \{ x_i \mid i=1 \text{ to } m \} \tag{2.1}$$

An *instance* x_i is a tuple of the form $(x^{(1)}, \ldots, x^{(n)}, y)_i$, where $x^{(1)}, \ldots, x^{(n)}$ represent values for the predictive attributes (features), while y is the value of the target attribute, the class:

$$x_i = (x^{(1)}, \ldots, x^{(n)}, y)_i \tag{2.2}$$

A *predictive attribute* X_i represents a recorded data characteristic, a *feature*, and the *class* Y is the target concept. The process of building the learning model is called *training* and is generally performed on a portion of the dataset, called the *training set*. In order to assess the expected value of the model its performance is measured on a *test set*, as we will see in section 2.3. For certain tasks, such as clustering, or outlier detection, the value of the target attribute may be unknown. This type of learning is referred to as *unsupervised learning*, as opposed to *supervised learning*, in which the class is used during model generation (e.g. classification). There exists also a special category of methods which perform *semi-supervised learning*, i.e. use for training a small amount of labeled data in conjunction with a large amount of unlabeled data. From a statistical perspective, X_i and Y are random variables; from the representation perspective, they can be either numeric, nominal or binary, or more complex data types, such as hierarchical, time series, etc. The range of Y calls for another distinction: if Y is continuous, then we are faced with a regression problem. On the other hand, if Y is discrete, we have a classification problem. A special case is binary classification, in which Y takes one of two values. The current thesis is concerned with classification tasks alone. We denote the cardinality of |Y| with c, and the possible values of Y with y^k. Therefore, $Y = \{y^1, \ldots, y^c\}$.

Summarization and visualization

This task performs initial data analysis. Descriptive statistics (mean, standard deviation, etc.) and data distribution visualization (through box plots, histograms, or 2D scatter plot) are the main alternatives for performing it.

Clustering

Clustering is involved with grouping the data into several clusters, by considering the similarity between individuals. The similarity computation is applied to the predictive attributes $X_1, \ldots X_n$ alone; the class attribute is unknown. The aim is to generate groups of individuals with a high degree of intra-cluster similarity and a low degree of inter-cluster similarity. Special attention should be allocated to the similarity metrics – they should be in accordance with the feature's nature and significance.

Link analysis

This task focuses on identifying relationships between data values. Two of the most common approaches are sequence and association mining. An association presents elements which co-occur in an event. It has the form *LHS=>RHS*, where *LHS* is a conjunction of terms from the set $\{X_1, \ldots, X_n\}$ and RHS contains another term from the set. Sequences are associations in time.

Outlier detection

Techniques for outlier detection generally build a model for identifying those instances in the dataset which deviate from normality. It is an unsupervised learning task, for which the training data is sampled from the "normal" instances.

Classification

Classification problems try to determine the characteristics which correctly identify the class to which each instance belongs to. Thus, the scope is to learn a model from the training set which describes the class *Y*, i.e. predict y from the values of (a subset of) $(x^{(1)}, \ldots, x^{(n)})$. The resulting model can be employed either for descriptive, or predictive tasks. Classification is similar to clustering, the main difference being that, in classification, the class to which each instance in the dataset belongs to is known a priori.

The most intense research efforts in DM and connected fields (machine learning, statistics) have focused on finding efficient classification algorithms, such that at the present there is a large collection of state-of-the-art methods available in the literature. This thesis focuses on DM classification tasks.

Regression

Regression is similar to classification, the difference being that the resulting model is able to predict numeric values – i.e. the class is numeric. There exist classifiers which can be used for regression as well (artificial neural networks, some decision trees).

Time series prediction

This task involves the prediction of future values from a series of time-varying predictors. Thus, $(x^{(1)}, \ldots, x^{(n)})$ represent values recorded (at certain time intervals t_1 to t_n) of the same measure, and *y* represents the predicted value for t_{n+1} . Like regression, time series prediction employs past values for prediction. The models must consider different properties of the time periods, especially seasonality and holidays, and must also establish how much of the past recordings is relevant.

2.3 Classification Methods

A wide variety of classification methods have become available during the last three decades. Among them, the most utilized general-purpose methods can be grouped into several categories (Table 2-1). Classification algorithms employ different theories for generating the model. The distinction between them is performed at the following levels: model representation and assessment strategies, form of the separation frontier, or model search techniques.

Table 2.1: The main categories of classification methods

Group of methods , with representatives	Model representation	Model evaluation	Model search	Separation frontier
Inductive methods: - decision trees: C4.5, CART,	Decision tree (interpretable)	Probabilistic methods, penalty on complexity	Greedy	Box –like
Coverage methods: - decision rules: OneR, PART, RIPPER	Decision rules: ranked disjunction of conjunctive rules (interpretable)	Classification error	Greedy	Similar to that of decision trees
Bayesian methods: - Bayes Net	Probabilistic, graphical (interpretable)	Likelihood, or posterior probability	EM	N/A
Linear methods[1]: - logistic regression, Fisher's Linear Discriminant, Support Vector Machines	Separation hyperplane (non-interpretable)	Likelihood, error, geometric margin	SMO, metoda punctului interior	Linear
Non-linear methods: - artificial neural networks	Network of interconnected neurons, links with weights (non-interpretable)	Mean squared error, cross entropy	Gradient descent, conjugate gradient, Newton's method, simm. annealing, EM, evolutionary, etc	Non-linear – depends on the activation function employed
Lazy methods: - k-Nearest Neighbor	Implicit (in data); no explicit model (non-interpretatable)	Classification error	External, non-specific to algorithm	Non-linear, irregular
Meta-techniques: - ensemble: bagging, boosting, wagging; stacking; multiple models techniques	Group of models, one for each base classifier (non-interpretatable)	Specific to the base classifiers	Specific to the base classifiers, boosting	Specific to the base classifiers

The main consequence of this fact is formulated by the *no-free lunch theorem*, which essentially states that there is no universally best classification method, which can achieve superior performance to all the others on any problem. Table 2.1 summarizes the specific choices made by different types of classification algorithms.

In addition, the classification algorithms and/or their resulting models possess several particularities, which can be evaluated when determining their suitability for a specific context: robustness to noise, scalability, speed (prediction speed mostly, but – in certain scenarios – training speed also), model interpretability, robustness to irrelevant/redundant features, robustness to numeric features, etc. These meta-characteristics are important for making the decision on which classifiers to consider initially for a certain application domain, given the specific constraints.

2.4 Model Evaluation

Many learning algorithms have become available to the community during the last few decades. However, according to the *no-free lunch theorem*, there is no "winner" method, which achieves superior performance to all the other methods, on all problems. Thus emerges the need for systematic procedures and measures which quantify performance and allows the comparison of the algorithms between each-other in different settings.

[1]Linear methods can be converted into non-linear, operating in a different input space, by using the kernel trick

2.4.1 Evaluation Tactics

The main difficulty in predicting the expected classifier performance on a new problem is the limited amount of data available and the fact that the sample may not be representative enough. Therefore, performing a single train-test split on the data, generating the model on the training set and evaluating its performance on the test split is, in most situations, insufficient, since it may provide a significantly biased estimation. Several different learner evaluation strategies are available in literature. All involve averaging the performance over several train-test iterations. The main idea in predicting the expected performance of a model is to do this on a new sample of instances, which was not "seen" by the learner during training. This is because estimations performed on the training set provide over-optimistic values.

The *de facto* standard in classifier evaluation is *k-fold stratified cross-validation*, in which the available data is divided into *k* folds of approximately the same size, each having the same distribution of instances as the original data. A number of *k* train-test trials are performed. In each trial, *k-1* folds are kept for training and one for testing, each time using a different fold for testing. The community is still debating on which value for *k* is the most appropriate, considering the time – accuracy of the estimation trade-off. The most frequent value is 10, but values of 2 and 5 are also very common. A single k-fold cross-validation might not provide a sufficiently reliable performance estimate, because of the effect of the random variation in choosing the folds. Therefore, it is common practice to repeat the process, usually 10 times. Thus, obtaining a good estimate of the performance of a classifier is a computation- and time-intensive task.

Leave-one-out cross-validation, or *m-fold cross-validation*, involves *m* trials; in each trial, the classifier is trained on *m-1* instances and tested on the remaining one. The main advantages of this approach are the large volume of data that can be employed for training and the fact that, since the procedure is deterministic, there is no need to repeat it several times. Unfortunately, the computational cost of executing it even once is high, since the learning-evaluation process has to be repeated *m* times. For large datasets this procedure is therefore unfeasible. Another drawback is the impossibility to ensure stratification. However, for small datasets, this method may provide a serious alternative to k-fold stratified cross-validation.

Bootstrap evaluation is based on the sampling with replacement statistical procedure. It applies the sampling strategy to the available data to generate the training set. Because some instances will be repeated in the resulting set, there will be a set of instances from the original data that do not appear in the training set. These will be employed for testing. It is known that the bootstrap estimate is a pessimistic one [Wit05]. Therefore, the output estimate is a linear combination of the bootstrap error and the error on the training set. The procedure is repeated several times, using different replacement samples, and the results are averaged.

2.4.2 Metrics

Almost all performance metrics are represented in terms of the elements of the confusion matrix generated by the model on a test sample. Table 2.2 presents the structure of a confusion matrix for a two-class problem, with classes *positive* and *negative*. A column represents an actual class, while a row represents the predicted class. The total number of instances in the test set is represented on the top of the table (*P=total number of positive instances*, and *N=total number of negative instances*), while the number of instances predicted to belong to each class are represented to the left of the table (*p= total number of instances classified as positive*; *n=total number of instances classified as negative*). TP (*true positives*) is the number of correctly classified positive examples.

Table 2.2: The confusion matrix returned by a classifier

Total no. of instances			**P**	**N**
			Actual class	
			positive	negative
p	**Predicted class**	positive	**TP**	**FP**
n		negative	**FN**	**TN**

In a similar manner, FN (*false negatives*) is the number of positive examples classified as negative, TN (*true negatives*) – the number of correctly classified negative examples and, finally, FP (*false positives*) – the negative examples for which the positive class was predicted.

The *true positive rate* (TP_{rate}): TP_{rate} = TP/(TP+FN) represents the rate of recognition of the positive class. It is also known as *sensitivity* (in ROC curves) and *recall* (in information retrieval). The corresponding measurement for the negative class is *the true negative rate* (TN_{rate}), also called *specificity*, and is computed as the number of negative examples correctly identified, out of all negative samples: TN_{rate}=TN/(TN+FP). In many applications, it is important to assess also how many examples which are identified as belonging to a given class actually belong to that class. This is done by the positive and negative predicted values. The *positive predicted value* (PPV), named also *precision* in information retrieval, is given by the number of actual examples identified as positive, out of all classified as positives: PPV=TP/(TP+FP), while the *negative predictive value* (NPV) represents the number of negatives correctly identified out of all examples classified as negative, NPV=TN/(TN+FP). The TP_{rate}, TN_{rate}, PPV and NPV indicate some true occurrences, which need to be maximized; sometimes their complements are more interesting. Their definitions may be found in [Wei03]. All these parameters provide a more exact view on the performance of a classification method. However, it is difficult to focus on all at the same time. Therefore, we should identify a connection between the problem objective and a subset of these eight measurements, and focus on those alone, or provide a composite metric which serves the given objective the best.

The most widely employed such metric in the early (theoretical) stage of data mining research was the *accuracy* (Acc) of the classifier, or its complement – the *error rate* (Err). Acc is defined as the ratio between the total number of correctly classified instances and the total number of instances, Acc = (TP+TN)/(TP+TN+FP+FN), while Err is the ratio between incorrect classified instances and all of the instances, Err = (FP+FN)/(TP+TN+FP+FN) = 1-Acc. They are usually expressed as percentages. The disadvantage of using these metrics is that they do not offer a realistic estimation when the distribution of instances is imbalanced. For example, for a problem in which 99% of the instances are negative and 1% positive, a model which always estimates the negative class achieves 99% accuracy, while failing to identify any positive instance. Such imbalanced distributions are common in real world scenarios, as we will see in Chapter 7 of this thesis.

Another such composite metric can be derived from the analysis of *ROC curves* [Faw06], which are graphs representing the TP_{rate} as a function of FP_{rate} (or, equivalently, they represent *sensitivity* as a function of *1-specificity*). A model built by a classifier generates a single point in this representation. A good model should be situated in the upper-left region of the chart. Point (0,0) represent a model which identifies all instances as being negatives, (1,1) one which identifies all as positives, while a random classifier lays on the y=x curve. The ideal model generates the point (0,1), meaning that FP_{rate}=0, (no negative instance identified as positive), and TP_{rate} =1 (all positives are identified). While this is the ideal case, very seldom a classifier can reach this performance. The ROC curve for a

classifier is built by generating several models (i.e. points in the ROC space). This can be done by repeatedly modifying the decision threshold of the classifier. ROC curves allow for the comparison of several classifiers. However, by simply analyzing the ROC curve, one can only get an approximate comparison. A more exact measure is provided by the *area under the curve (*AUC). It is a scalar measure which should be maximized. However, in many situations we cannot identify a single dominating curve, but intervals in which different curves dominate all the others. This means that the curve yielding maximal AUC might not be the most appropriate under all circumstances. For instance, a curve with smaller AUC dominates the curve with highest overall AUC on a narrow interval, i.e. is more appropriate for that set of requirements. In such complex cases, for the same problem, based on the specific problem requirements and objectives, different classifiers might be preferable for each set of constraints, as suggested by the convex hull methodology [Pro97]. This requires a more sophisticated approach, which is not so easy to follow.

A good option for when both recall and precision are important is the *F-value* [Guo04], which is the harmonic mean between precision and recall; a generalization of the metric – the F_i*-value* – can be tuned to put more emphasis on either the recall or precision:

$$F_i\text{-}value = (1+i^2) \cdot precision \cdot recall/(i^2 \cdot recall+precision) \quad (2.3)$$

When we need to accentuate recall more, *i* should be greater than 1. For a specific problem, the goal is to identify the appropriate *i* such that the proper amount of penalization for the false negatives is provided [Est01].

In certain situations, besides the TP_{rate}, keeping a high TN_{rate} may be important. For such situations, equidistant metrics, such as the *geometric mean*: GM = $\sqrt{TP_{rate} * TN_{rate}}$ [Bar03], or the *balanced accuracy*: BAcc = $(TP_{rate}+TN_{rate})/2$ [Bro10] are appropriate performance assessment metrics.

The choice of the performance metric should be influenced strongly by the specifics of the problem at hand. Any performance metric may be perfectly appropriate in a given scenario, yet fail to provide a real estimation on the expected performance in a different problem setting. For example, in medical diagnosis, it is essential to maximize the TP_{rate}, even if this means that a certain number of FPs are introduced. On the other hand, in contextual advertising, precision is of utmost importance, since it is more important that the ads predicted as being relevant are actually relevant, than to identify as many relevant ads as possible. Therefore, selecting the appropriate performance metric, which is in accordance with the specific problem goals, is essential in reducing the risks of a failed data mining process.

3 Handling Incomplete Data

Incomplete data has proved to be the rule rather than an exception in real-world data mining problems. At the same time, it represents a challenge for achieving a successful data mining process, since statistical and machine learning methods have not been designed to deal with incomplete data, and most of them employ simple and inefficient approaches: consider all missing values as a special value, replace all missing values with NULL, remove all instances/attributes having missing values. A limited number of learning algorithms, such as the C4.5 decision tree learner, apply slightly more evolved methods. Adequate handling of missing data in the pre-processing phase, for improving the quality of raw data and thus reducing the risk of a failed DM process, is therefore essential.

3.1 Problem Statement

An **incomplete instance** is characterized by unknown values for at least one of the predictor attributes X_p:

$$(x_j)_{inc}=(x^{(1)}, \ldots, x^{(p-1)}, ?, x^{(p+1)}, \ldots, x^{(n)}, y)_j,$$

where "?" denotes an unknown value for attribute X_p. Of course, an instance may have several unknown values.

In selecting the appropriate approach for handling incompleteness, knowledge about the mechanisms which generated the incomplete data, as well as about incompleteness models – if avaliable – may provide valuable support. In [Rub87b], the incompleteness mechanism is modeled probabilistically, by defining (for the training set T) the *incompleteness matrix* M:

$$m_{ij} = \begin{cases} 1, if\ x_{ij}\ is\ absent \\ 0, if\ x_{ij}\ is\ recorded \end{cases} \quad (3.1)$$

The form of the matrix M characterizes the incompleteness model. The three most encountered patterns are:

- univariate non-response – in which a single attribute X_i is incomplete
- monotone – in which if $x_j^{(i)}$ is missing (i.e. $x_{ij} = 1$), then $x_j^{(i+1)}$, ..., $x_j^{(n)}$ are also missing; otherwise said, the 1s in the matrix form a stair-like pattern
- general (arbitrary) – the matrix M has no special form

The incompleteness mechanism, i.e. the law generating the missing values is crucial, since, by knowing it, missing values could be estimated more efficiently. It is characterized by the conditional distribution of M, given T – $f(M \mid T, \phi)$ – where ϕ represents unknown parameters. If we consider $T = T_{obs} + T_{inc}$, i.e. decompose the training set into a complete component and an incomplete component, we distinguish the following mechanisms [Rub87b]:

- **MCAR** („Missing Completely At Random"): the incompleteness probability density function does not depend on the data in any way:
  $$f(M \mid T, \phi) = f(M \mid \phi),\ \forall\ T, \phi$$
- **MAR** („Missing At Random"): the incompleteness probability density function may depend on the recorded data, but not on the incomplete data:
  $$f(M \mid T, \phi) = f(M \mid T_{obs}, \phi),\ \forall\ T_{inc}, \phi$$
- **NMAR** („Not Missing At Random"): the missing probability depends on the missing value:
  $$f(M \mid T, \phi) = f(M \mid T_{obs}, T_{inc}, \phi),\ \forall\ \phi$$

To make the differences between the missingness mechanisms more clear, suppose that, in the available data for a medical diagnosis problem, each record contains the patient id, age, sex and the value of a medical test. Some values for the medical test are missing. If the fact that the test result is missing doesn't depend on the recorded or missing data (i.e. the test is expensive, so it is performed only on a subset of patients, randomly chosen), then the data are MCAR. If, on the other hand, the test is mainly performed on elder females, then the data are MAR (the fact that the test result is missing depends on the observed variables age and sex). If the missing data comes as a result of the fact that the device used to perform the test is not able to record values above some threshold, then the data are NMAR. This formalism is particularly important for statistical inference [Rub87b]. MCAR ang MAR provide ignorable incompleteness: since the missingness is not related to the missing values, it can be ignored. NMAR data is informatively missing, which means not only that the incompleteness mechanism is non-random and cannot be predicted from the other variables in the dataset, but also that it cannot be ignored. In practice, it is generally very difficult to determine the incomleteness mechanism. Data can generally provide evidence against MCAR. However, it cannot, in general, distinguish between MAR and NMAR, without making distributional assumptions.

For classification problems, a very important distinction should be made for the incompleteness mechanisms – whether they are:

- **Informative**: the fact that the value is missing provides information about the target attribute
- **Non-informative**: missing value distribution is uniform for all target attribute values

This distinction is important since informative incompleteness should be treated as a distinct value, because it provides additional information about the target attribute. For non-informative incompleteness, imputation methods are appropriate. Also, most embedded mechanisms for handling incompleteness are appropriate only for non-informative incompleteness [Sch04]. The two incompleteness mechanisms typologies – MAR/MCAR/NMAR and informative/non-informative – overlap somehow, even if the former targets estimation methods for statistical density/models (where the existence of a target concept) is not mandatory. However, we can state that: MCAR are always non-informative; MAR can be both informative and non-informative, depending on the dataset it applies to – whether it contains the target attribute or not; for NMAR, a value is (potentially) informative if the true value is statistically dependent on the target attribute, and non-informative otherwise.

3.2 Methods for Dealing with Missing Data

Missing data techniques (MDTs) are available either as pre-processing methods or as embedded approaches – several learning methods possess specific strategies for handling incompleteness. Pre-processing methods build a complete dataset and are preferred since the resulting data can be subsequently analyzed using any of the traditional learning methods. The main strategies which apply here are *filtering* and *imputation*. It is worth mentioning that not all MDTs are equally appropriate for estimation or prediction tasks [Sar98].

3.2.1 Filter-based MDTs

Traditionally, filtering methods simply remove the incomplete data parts: *list-wise deletion (or complete-case analysis)* deletes all incomplete instances; it is therefore appropriate only when there are few incomplete instances, whose influence on the rest of the population is minimal; *pair-wise deletion (or available-case analysis)* applies a more efficient approach for rare information – whether a particular case is missing or not depends on the particular goal of the analysis. These methods have been the most common approaches

for handling missing data for several years, despite their evident drawbacks: they generally induce bias in the subsequent analysis (if the data is not MCAR, which is seldom the case), and they do not maximize the use of available data [All09].

More evolved methods which employ filtering as part of their strategy include: the *multiple models* and the *sub-model derivation* methods [Sch04]. The *multiple models* approach splits the available data into complete sub-sets and generates learning models on each sub-set. The main disadvantage of this method is represented by the combinatorial explosion of sub-sets for spaces with many attributes and different incompleteness mechanisms. Also, some sub-sets may be too small for inferring acceptable models. Prediction combination and the existence of several appropriate models for certain cases also represent challenges for this approach. *Sub-model derivation* is similar with the multiple models approach, but avoids the combinatorial models explosion; the condition is the possibility to derive a sub-model for any attribute sub-set from the model on the complete attribute space. This way, the complete model is built only once, and whenever we have to make a prediciton for an incomplete instance, we derive the appropriate sub-model from the complete one. For building the complete model, the strategy can be extended such as to allow the adjustment of the complete model whenever a sub-model changes.

3.2.2 Imputation-based MDTs

The main assumption employed by imputation-based methods is related to the existence of correlations between the incomplete variables and the other variables in the dataset. For each missing item, a replacement value is derived from the available data, such as to produce a complete dataset. There exist several different strategies for devising imputation-based MDTs. Imputations may be *deterministic* (i.e. by repeating the process you always get the same replacement value) or *stochastic* (i.e. randomly drawn from some distribution). Depending on the type of information used to derive replacement values, imputation methods are either based on the *missing attribute* or based on *non-missing attributes*; depending on the amount of data used, MDTs may be *global* or *local*. The most appropriate classification of imputation-based methods is the one proposed in [Mag04] – which we further employ to present imputation-based MDTs.

Global imputation based on missing attribute

This category of methods assigns values to an incomplete attribute by analyzing the existing values for that attribute. The most common approaches are *mean*, *median*, or *mode imputation*. Their major drawback is the fact that they bias standard deviation estimations, even for MCAR data. The *non-deterministic mean imputation* method, which introduces a random disturbance to the mean, achieves better performance from a statistical point of view, but is still not satisfactory for learning tasks.

A variation of this method is to assign *all possible values* to the missing data item. This produces an inconsistent dataset, but the biggest problem with the approach is its computational complexity and the fact that it systematically introduces information to the data.

Global imputation based on non-missing attribute

The main assumption employed by global imputation based on non-missing attribute methods is the existence of correlations between missing and non-missing variables. They employ these correlations to predict missing values. *Imputation by regression* is one such method, which treats missing attributes as the target attribute, in turn, and performs regression to impute missing values. In [Mag04], it is noted that this strategy possesses two main problems: *(1)* the selection of the appropriate type of regression model is crucial (e.g. if linear regression is performed but the data doesn't fit a linear model, erroneous values are

imputed) and *(2)* only one value is imputed for each missing data and this fails to represent correctly the uncertainty associated with missing values.

Multiple imputation, or *MI*, [Rub87] comes as a solution to the second issue stated above. It consists of 3 steps: *imputation* – create *m* complete datasets, through single imputation; *analysis* of each of the *m* datasets; *combination*, by integrating the results of the *m* analyses into the final result. Even if MI appears to be the most appropriate method for general-purpose handling of incomplete data in multivariate analysis, in practice, the requirements which have to be met in order to achieve MI's desirable properties, are normally violated: the data is not generally MAR, and producing random imputations that yield unbiased parameter estimates is not a trivial task.

Local imputation

A large number of local imputation methods are grouped under the name of *hot-deck imputation.* It is a generic procedure, which doesn't have a strong theoretical formulation, but has been implemented in practice in several different versions [Grz02, Bat03, Ged03]. For each value to impute, it:

1. finds similar records with the current record
2. use a certain strategy to derive the replacement value from the corresponding values of the neighboring records

The assumption al hot-deck procedures are based on, that instances can be grouped in classes, with small variation within each class, is generally verified in DM tasks [Mag04]. Exceptions may appear for attributes which don't exhibit any (or a very low) correlation with the other attributes in the dataset.

Imputation via parameter estimation

Imputation values may be deduced also by first estimating the parameters of the multivariate probability density function and then using the resulting data model to impute the missing portion of the data. *Expectation Maximization (EM)* is such an approach, which uses maximum likelihood to estimate the parameters of a probabilistic model which best explain the data. For incomplete data, missing values represent hidden variables. It is an iterative method, which performs two operations in each iteration: (1) the *E step* – compute the expectation of the likelihood, using the distribution of the hidden variables, which can be derived from the current parameter estimations; (2) the *M step* – maximize similarity relative to the parameters. The main drawbacks of the approach are related to the computational complexity and the fact that in real DM problems it is generally difficult to guess the probability density function in advance.

3.2.3 Embedded MDTs

Several decision tree classification methods possess non-trivial methods for handling incomplete data. In C4.5 [Qui93], for example, in the training phase, the gain ratio of each attribute is adjusted by a factor which depends on the number of complete records (in that attribute) in the training set. Every incomplete record is distributed among all partitions, with a probability which depends on the size of the partition (*fractional cases*). In the prediction phase, when a test on an unknown attribute has to be performed, the instance is again propagated on all available paths, on each branch having a weight that corresponds to the relative frequency of a value assigned to that branch.

The *CART* (*Classification and Regression Trees*) method [Bre84] applies the so-called *Surrogate Variable Splitting* (SVS) strategy. The method employs "surrogate splitters", i.e. predictor variables which yield similar splitting results with the primary splitter, to replace the primary splitter in nodes when there are missing values. This strategy can only be used in the prediction phase. The *RPART* (*Recursive Partitioning and Regression Trees*)

method [The97] contains an extension of the basic method, which can handle missing data during training as well.

3.3 A New Method for Data Imputation

Considering the superior performance achieved by ensembles of artificial neural networks on various problems [Ona07], a new imputation method, which applies the predictive capabilities of an ensemble of artificial neural networks to impute missing values, has been proposed. This section presents the technique, together with the empirical evaluations performed to validate the idea. The proposed method employs global imputation based on non-missing attributes. The strongest assumptions made by the method are related to the existence of correlations between the incomplete and the complete attributes (including the class). The particular technique employed for learning the model used to impute each incomplete attribute reduces the risk of choosing an inappropriate model. Also, even if there is a single imputation performed for each missing value, the replacement value is determined by an ensemble of predictors, using a voting strategy. Even if this doesn't improve the representation of the uncertainty associated with missing values, it reduces the risk of imputing wrong values.

3.3.1 Method Description

Before describing the actual technique, we should make a note on the fact that the purpose of the method is to improve the classification performance, i.e. the prediction capabilities of the model built on incomplete data. The technique implies training an artificial neural network ensemble model for each attribute, using the complete data sub-sets for the particular attribute, considered as target. For each predictor attribute X_j, $j = 1$ *to* n, from the available training data T two sub-sets are extracted:

- a complete data kernel – CT_j – which is employed to build the attribute I imputation model
- the X_j–incomplete subset – IT_j – which contains all instances with the value of attribute X_j missing.

For univariate incompleteness, for each attribute X_j and instance $x_i \in T$, $x_i \in CT_j$ if $x_i^{(j)}$ is known, and $x_i \in IT_j$ otherwise ($x_i^{(j)}$ = „?"). The artificial neural network ensemble is trained on CT_j, considering X_j as target attribute, and the resulting model is employed to impute the values of X_j in IT_j, obtaining IT_j'. The resulting training set possesses a higher quality and thus improves the quality of the resulting prediction model.

```
Procedure NNE_Impute (T,Xj)
(1)  Create Tc = T
(2)  for each instance xi∈Tc
(3)        if xi(j)="?" then
(4)              ITj= ITj ∪{xi(j)}
(5)           else
(6)              CTj=CTj ∪{xi(j)}
(7)        end if
(8)  end for
(9)  Model Mj= NNE_Train(CTj, Xj)
(10) for each instance xi∈ITj
(11)       Ik = Select corresponding instance from Tc
(12)      Set Ik=Mj(Ik)
(13) end for
(14) return Tc
```

The code snippet on the previous page presents imputation method for the univariate case – in which $T = IT_j \cup CT_j$ and $IT_j \cap CT_j = \varnothing$. For more complex incompleteness mechanisms, the difference is in the strategy for extracting the complete data subset.

3.3.2 Experimental Evaluation

The purpose is to show that, by using the imputation method, higher prediction performance is obtained than by using the incomplete dataset. The quality of the imputed values results indirectly, from the superior classification performance. We have performed several evaluations, considering univariate incompleteness with varying levels and measuring the classification accuracy of the J4.8 decision tree classifier – which possesses also embedded mechanisms for handling incomplete data. The incompleteness mechanism is MCAR.

For each attribute in turn, we have compared the performance of the models trained on:

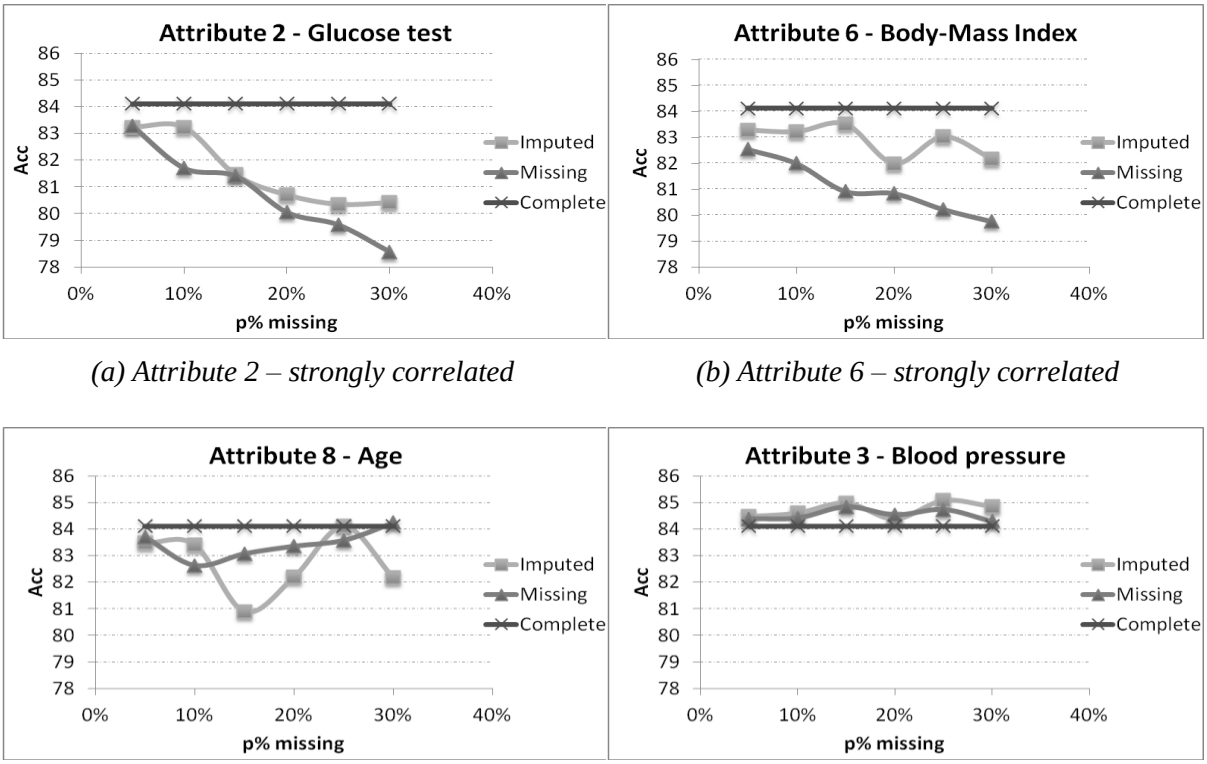
- The complete dataset: $IT_j = \varnothing$, $T = CT_j$
- The *p%* incomplete dataset: *p%* values for attribute X_j are artificially removed, using an MCAR strategy, with *p* varied between 5 and 30, using an increment of 5
- The *p%* imputed dataset: the same *p%* values for attribute X_j from before are imputed using our imputation method

The evaluations have been performed on two complete UCI datasets (Pima Indian Diabethes and Cars), using 10 different trials of 10-fold cross validation loops (100 runs in total). We have studied the impact of the imputed values on the performance of the prediction models. The relation set as goal between the three measures is:

Accuracy(complete train set) >
Accuracy(imputed train set) >
Accuracy (incomplete train set)

The diagrams (a) – (d) from figure 3.1 present the results obtained for two strongly, one moderately and one weakly correlated attribute with the class, for the Pima dataset. The results indicate that, for strongly correlated attributes with the class, the expected relationship between the accuracies on the different versions of the training set is correct. A moderately correlated attribute (attribute 8 – age) exhibits a relation close to the expected one, and the rest of the attributes achieve the highest classification accuracy when using missing or imputed values (which is a deviation from the expected behavior). Moreover, the curve representing the class accuracy when using missing values in the training phase should decrease as the percentage of the incompleteness increases. Again, only some attributes exhibit this behavior. If we further analyze the ranking of the attributes (using the gain ratio as ranking metric), we come across the following order: **2, 6,** *8*, 1, 5, 7, 4, 3. The complete attribute list for the Pima dataset can be found in table A.6.5 of Appendix A. For this dataset, which does not represent the problem well enough (since the accuracy is around 84%), only imputation on the attributes which are highly correlated with the class guarantee performance improvement.

A second batch of tests was conducted on the Cars dataset, which consists of nominal attributes, and yields a high accuracy on the complete training set (~96%). For this dataset (figure 3.2, diagrams (a) – (f)), almost all the attributes exhibit the normal relation between the three curves. Also, we can observe how the class accuracy decreases as the number of missing attribute values increases, which is the expected behavior. The attribute ranking for the Cars dataset is: **6**, **4**, *1*, *2*, *5*, 3.



(a) Attribute 2 – strongly correlated

(b) Attribute 6 – strongly correlated

(c) Attribute 8 – moderately correlated

(d) Attribute 3 – weakly correlated

Figure 3.1 - The variation of the accuracy with different incompleteness percentages for several attributes of the Pima dataset

Therefore, when the dataset represents well the search space, our method boosts the accuracy of learners (the accuracy when using the predicted values for the attributes is higher than the one achieved with incomplete data). Although the method has been evaluated for handling one missing attribute at a time, it can be applied to multivariate incompleteness, provided that a complete subset of instances can be extracted from the available training data. Then, imputation can be performed incrementally, first on instances having one missing attribute value, then two, and so on.

3.4 Conclusions on Data Imputation

The successful application of learning methods to real-world problems is frequently conditioned by aspects involving the quality of the available data, such as noise or incompleteness. MDTs are available either as pre-processing methods – filters and imputation methods – or as embedded approaches. Their success in a certain scenario depends on both the match between the assumptions made by the MDT technique and the incompleteness mechanism (which is data dependent) and on the subsequent processing applied to the data (for pre-processing methods). Moreover, filters tend to introduce bias in the learning process and reduce the volume of available data, while existing imputation methods may require strong assumptions which don't generally hold in real situations, or employ knowledge about the data distribution, which is not always accessible. Computational complexity may also represent an issue for some imputation techniques.

Having the high accuracy obtained by the ensemble of neural networks as motivation, a new global imputation method based on non-missing attributes has been proposed, implemented and evaluated on benchmark data, using univariate MCAR as incompleteness mechanism.

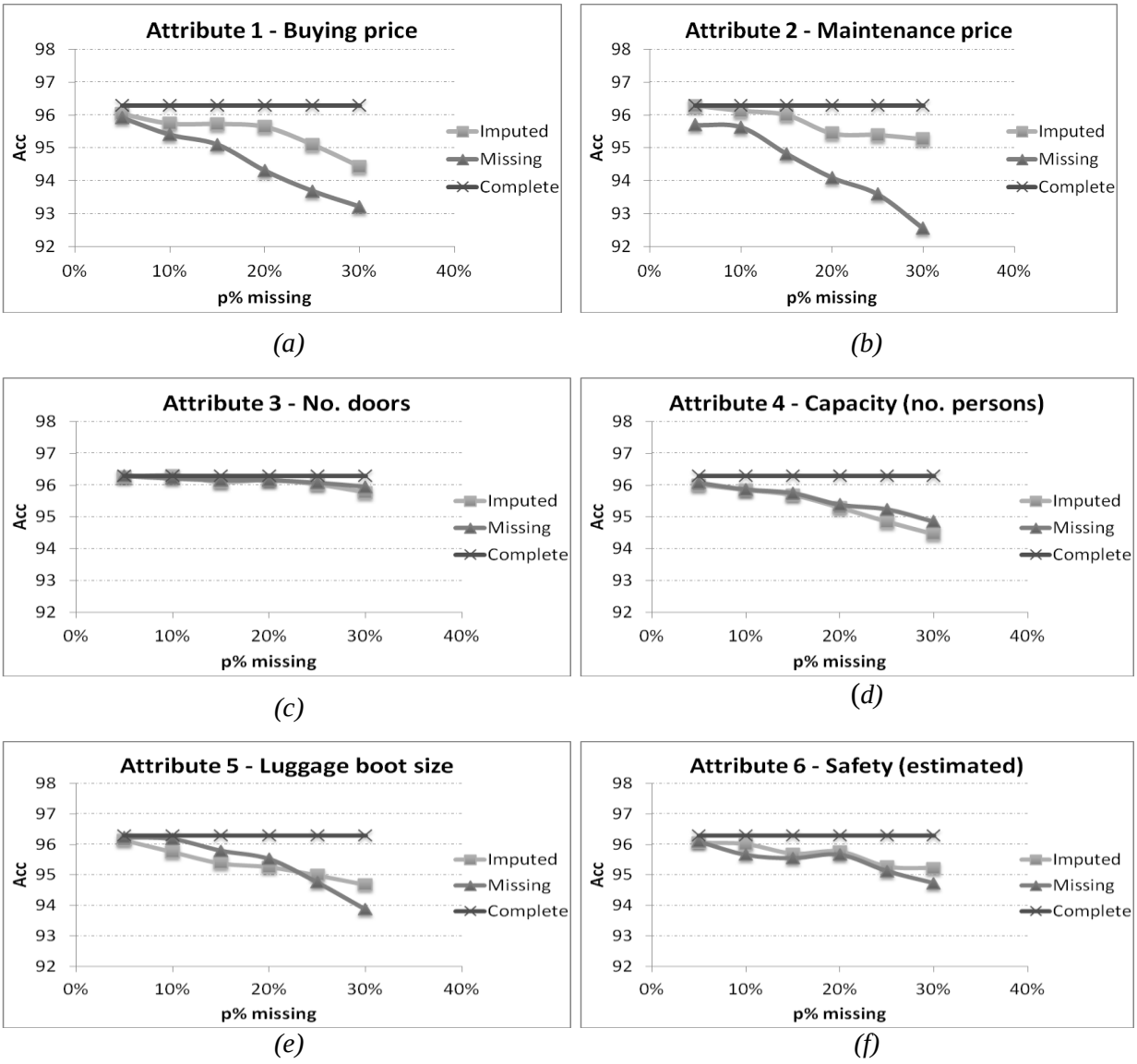


Figure 3.2 - The variation of the accuracy with different incompleteness percentages for the attributes of the Cars dataset

The results have shown that the method can improve the classification performance by imputing missing values, but not for all attributes in a dataset. Thus, for a dataset with a high level of accuracy (e.g. Cars, ~96% accuracy on complete set) improvements have been observed for almost all the attributes. But for the Pima dataset, on which classifiers achieve a lower accuracy (~84%), improvements have been observed only for the attributes strongly correlated with the class. The method can be extended to accommodate more complex incompleteness patterns – by performing incremental imputations – initially identify complete data kernels, and perform imputation incrementally, one attribute at a time.

The experiments have shown a strong connection between the correlation of a certain attribute with the class and the performance of our imputation method for that attribute. Therefore, considering feature selection information in the imputation step may boost the success of the imputation. Current research efforts focus on evaluating the method on other incompleteness mechanisms and patterns.

The original data imputation method proposed in this chapter is the result of research supported by PNII grant no.12080/2008: SEArCH – Adaptive E-Learning Systems using

Concept Maps. The results have been acknowledged by the scientific community through publication of 1 research paper, presented at an international conference:

1. **Vidrighin Bratu, C**., Muresan, T. and Potolea, R., "Improving Classification Performance on Real Data through Imputation", *Proceedings of the IEEE International Conference on Automation, Quality and Testing, Robotics*, 22-25 May, Cluj-Napoca, Romania, Vol. 3, pp. 464-469, ISBN: 978-1-4244-2576-1, 2008

4 Feature Selection

Dimensionality reduction through the selection of a relevant attribute (feature) subset may produce multiple benefits to the actual data mining step, such as: performance improvement, by alleviating the curse of dimensionality and improving generalization capabilities, speed-up by reducing the computational effort, improving model interpretability and reducing costs by avoiding "expensive" features. These goals are not fully compatible with each other. Thus, there exist several feature selection problems, according to the specific goals. In [Nil07], feature selection problems are classified into two main categories: finding the optimal predictive features (for building efficient prediction models) and finding all the relevant features for the class attribute.

From a purely theoretical perspective, the selection of a particular attribute subset is not of interest, since the Bayes optimal prediction rule is monotonic, hence adding more features cannot decrease accuracy [Koh97]. In practice, however, this is actually the goal of feature selection: selecting the best possible attribute subset, given the data and learning algorithm characteristics (such as biases, heuristics). Even if there exist certain connections between the attributes in the subset returned by some techniques and the theoretically-relevant attributes, they cannot be generalized to form a practical methodology, applicable to any learning algorithm and dataset. This is because the information needed to compute the degree of relevance of an attribute (i.e. the true distribution) is not generally available in practical settings.

4.1 Problem Statement

The concept of ***relevance*** is central to the theoretical formulation of feature selection. There are several definitions of relevance available in literature. In [Gen89], a feature is defined as relevant if its values vary systematically with the class attribute values. In [Koh97], this is formalized as:

Definition 1: X_j is **relevant** *iff*
$\exists$ *x and y for which* $p(X_j = x)>0$, *s.t.* $p(Y = y \mid X_j = x) \neq p(Y = y)$,

meaning that an attribute is relevant if the class attribute is conditionally dependent on it.

Another possible definition of relevance is that by removing attribute X_j from the feature set F the conditional class probability changes [Koh97]:

Definition 2: X_i is **relevant** *iff*
$\exists$ *x, y and f for which* $p(X_j = x, F_j = f)>0$, *s.t.* $p(Y = y \mid X_j = x, F_j = f) \neq p(Y = y \mid F_j = f)$
where $F_j = \{X_1, \ldots X_{j-1}, X_{j+1}, \ldots X_n\}$ denotes the set of all attributes except X_j and f represents a value assignment to F_j.

However, these definitions may yield unexpected results. Take the Boolean `xor` problem, for example, with $Y = X_1 \oplus X_2$. Both X_1 and X_2 are indispensable for a correct prediction of Y. However, by the first definition, both X_1 and X_2 are irrelevant, since $p(Y=y \mid X_1 = x) = p(Y=y) = 0.5$, i.e. for any value X_1 there are two different values for y. The same is true for X_2. Also, if we add feature $X_3 = \neg X_2$, then by the second definition, both X_2 and X_3 are considered irrelevant, since neither adds information to F_3 and F_2, respectively.

To address such issues, in [Koh97] two degrees of relevance are introduced: **strong relevance** and **weak relevance**, by quantifying the effect of removing the attribute on the performance of the Bayes optimal classifier. Thus, an attribute is strongly relevant if it is

indispensable, i.e. its removal results in performance loss of the optimal Bayes classifier. The actual definition is equivalent to the second definition of relevance presented before.

The definition for weak relevance is the following:

Definition 3: *An attribute X_j is **weakly relevant** iff it is not strongly relevant and $\exists$ a subset of features F_j' of F_j for which $\exists$ x, y and f' with $p(X_j = x, F_j' = f') > 0$, s.t.*

$$p(Y = y \mid X_j = x, F_j' = f') \neq p(Y = y \mid F_j' = f')$$

A feature is relevant if it is either strongly or weakly relevant, and irrelevant otherwise. For the `xor` problem, X_1 is strongly relevant, and X_2 and X_3 are weakly relevant.

The above definitions of relevance do not imply attribute usefulness for a **specific** learner. Therefore, we define feature selection in the following manner:

Definition 4: Feature selection represents the extraction of the **optimal** attribute subset**,**

$$F_{opt} = \{X_{k1}, ..., X_{kn}\}, \text{ where } \{k_1, ..., k_n\} \subseteq \{1,...,n\}$$

The definition of optimality is specific to the feature selection technique (on the *subset evaluation measure*), and it depends on the learning algorithm characteristics (such as biases, heuristics) and on the end goal of the classification.

4.2 Feature Selection Techniques

There exist two possible strategies to follow for feature selection: search for the best subset of predictive features (for building efficient prediction models), or find all the relevant features for the class attribute. The latter is achieved by performing a ranking of the attributes according to their individual predictive power, estimated via different tactics: (*i*) compute the performance of a classifier built with each single variable, (*ii*) compute statistic measures, such as a correlation coefficient or the margin and (*iii*) employ information theory measures, such as the mutual information [Guy03]. However, this approach fails to identify redundant features, which have been shown to harm the classification process of the naïve Bayes classifier [Lan94a]. Therefore, most feature selection techniques focus on searching for the best subset of predictive features. They differ in two important aspects: the search strategy employed and the attribute subset evaluation method.

There exist several comprehensive studies on feature selection algorithms in literature. Dash [Das97] classifies feature selection algorithms using two criteria: the generation procedure and the evaluation function. Three generation procedures – heuristic, complete and random – and five evaluation measures – distance, information, dependency, consistency and classifier error rate – are identified. This results in fifteen possible combinations; the representative algorithms in each class are reviewed. Empirical evaluations are performed on three artificial datasets, to study the capacity of each algorithm to select the relevant features. Another comprehensive survey is presented in [Mol02]. The classification of feature selection algorithms presented there is similar to that in [Das97]. The difference is that the generation procedure is further divided into search organization and generation of successors, resulting in a 3-dimension characterization of the feature selection methods. Other valuable studies can be found in [Guy03, Nil07].

Feature selection algorithms are traditionally divided in machine learning literature into: filter methods (or filters), wrapper methods (or wrappers), and embedded methods (i.e. methods embedded within the learning process of certain classifiers).

4.2.1 Search Strategies

For the feature selection problem, the order of the search space is $O(2^{|F|})$. Therefore, performing an exhaustive search is unfeasible except for domains with a small number of features. Complete search strategies perform a complete search for the optimal subset, according to the evaluation function used. Their complexity is smaller than $O(2^{|F|})$, because not all subsets are evaluated. The optimality of the solution is guaranteed. Representatives of this category are: branch and bound with backtracking or breadth-first search.

A more efficient trade-off between solution quality and search complexity is provided by heuristic search procedures. With a few exceptions, all search strategies falling in this category follow a simple process: in each iteration, the remaining features to be selected/rejected are considered for selection/rejection. The order of the search space for these procedures is generally quadratic in the number of features – $O(|F|^2)$. Therefore, such methods are very fast, and although they do not guarantee optimality, the quality of the solution is usually good. Representative of this category are: greedy hill climbing, which considers local modifications to the feature subset (forward selection, backward elimination or stepwise bi-directional search), best-first search, which also makes local changes but allows backtracking along the search path and genetic algorithms, which consider global changes. Given enough time, best-first search will perform a complete search. Greedy hill-climbing suffers from the horizon effect, i.e. it can get caught at local optima. Given enough search variability, population size and number of iterations, genetic algorithms usually converge to the optimal solution.

A less investigated strategy is random search, which limits the number of evaluated subsets by setting a maximum number of iterations possible. The optimality of the solution depends on the resources available and adequate values for certain parameters. Representative for this category is the Las Vegas search algorithm. Also, a certain degree of randomness can be found in genetic algorithms and simulated annealing; greedy hill climbing can be injected with randomness by starting from an initial random subset.

4.2.2 Evaluation Measures

Whether or not a certain sub-set is optimal depends on the evaluation criterion employed. The relevance of a feature is relative to the evaluation measure.

Since, most often, the end goal of feature selection is to obtain an efficient classification model in the processing phase, setting the target of feature selection to minimize the (Bayesian) probability of error might be appropriate. The probability of error is defined as [Dev82]:

$$P_e = \int [1 - \max_k P(y^k \mid x)] p(x) dx \tag{4.1}$$

where $p(x) = \sum_{k=1}^{c} p(x \mid y^k) P(y^k)$ is the unconditional probability distribution of the instances, and $P(y^k \mid x)$ is the posterior probability of y^k being the class of x.
The goodness of a feature subset F' is therefore: $J = 1 - P_e$. $P(y^k \mid x)$ are usually unknown, and have to be modeled, either explicitly (via parametric or non-parametric methods), or implicitly, by building a classification model which learns the decision boundaries between the classes on a sample dataset. For a feature subset F', an estimate $\hat{P}_e$ of the error is computed, by counting the errors produced by the classifier built on a subset of the available data, using only the features in F', on a holdout test set taken also from the available data. The feature subset which minimizes the error is returned. This forms the basis of the wrapper methodology. The estimation of $\hat{P}_e$ may require more sophisticated procedures than simple

holdout validation set: k-fold cross-validation or repeated bootstrapping may yield more accurate values.

Distance (or discrimination, separability) measures favor features which induce a larger distance between instances belonging to different classes. The Euclidean distance is one of the metrics used to compute the distance. The best feature subset is that which maximizes the inter-class distance [Mol02]:

$$J = \sum_{k=1}^{c} P(y^k) \sum_{l=k+1}^{c} P(y^l) D(y^k, y^l) \tag{4.2}$$

where y^k and y^l represent the k^{th} and l^{th} class labels, respectively, and

$$D(y^k, y^l) = \frac{1}{N_k N_l} \sum_{k_1}^{N_k} \sum_{k_2=k_1+1}^{N_l} d(x_{(k,k_1)}, x_{(l,k_2)}) \tag{4.3}$$

represents the interclass distance between the k^{th} and l^{th} class labels, N_k and N_l are the number of instances belonging to classes N_k and N_l, respectively, and $x_{(k,k1)}$ is the instance k_1 of class y^k. Such measures do not necessitate the modeling of the probability density function. As a result, their relation to the probability of error can be loose [Mol02].

Divergence measures are similar to distance measures, but they compute a probabilistic distance between the class-conditional probability densities:

$$J = \int f[p(x \mid y^k), p(x \mid y^l)] dx \tag{4.4}$$

Classical choices for *f* include: the Kullback-Liebler divergence or the Kolmogorov distance. Such measures provide an upper bound to P_e.

Statistical dependence quantify how strongly two features are associated with one another, i.e. by knowing the value of either one, the other can be predicted. The most employed such measure is the correlation coefficient:

$$Correlation(X_i, X_j) = \frac{E[(X_i - \mu_{X_i})(X_j - \mu_{Xj})]}{\sigma_{X_i} \sigma_{Xj}} \tag{4.5}$$

where μ represents the expected values and σ standard deviations. The correlation can be estimated from a data sample, i.e. the training set:

$$r_{x^{(i)}x^{(j)}} = \frac{\sum_{k=1}^{m} (x_k^{(i)} - \overline{x^{(i)}})(x_k^{(j)} - \overline{x^{(j)}})}{(m-1) s_{x^{(i)}} s_{x^{(j)}}} \tag{4.6}$$

where $x^{(i)}$and $x^{(j)}$ represent the value sets of attributes X_i and X_j, respectively, $\overline{x^{(i)}}$ and $\overline{x^{(j)}}$ represent the sample means and $s_{x^{(i)}}$ and $s_{x^{(j)}}$ represent the sample standard deviations.

This measure may be used in several ways: rank features according to their individual correlation with the class – those exhibiting a large correlation are better; a second possibility is investigated in [Hal00], where the heuristic "merit" of a subset of features is proposed, according to which subsets whose features exhibit higher individual correlation with the class and lower inter-correlation receive higher scores:

$$M_{F'} = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \tag{4.7}$$

where $k = |F'|$, r represents the sample correlation coefficient, c represents the class and f represents a predictive feature; $\overline{r_{cf}}$ is the mean feature-class correlation, $f \in F'$and $\overline{r_{ff}}$ is the mean feature-feature inter-correlation.

Similar to the statistical dependence measures, there are several measures from information theory, based on Shannon's entropy, which can help determine how much information on the class Y has been gained by knowing the values of X_i. The most employed is the information gain, which can be used without knowledge of the probability densities, such as in decision tree induction.

Consistency measures are characteristically different than all the other evaluation measures. They rely heavily on the training data. Also, the methods that employ them apply the Min-Features bias, i.e. favor consistent hypotheses definable over as few features as possible. An inconsistency in *F'* appears when two instances belonging to different classes are indistinguishable by their values of the features in *F'* alone. The inconsistency count of an instance x_i with respect to feature subset *F'* is:

$$IC_{F'}(x_i) = F'(x_i) - \max_k F'_k(x_i) \tag{4.8}$$

where $F'(x_i)$ is the number of instances in training set *T* equal to x^i using only attributes in *F'*, and $F'_k(x_i)$ is the number of instances in T of class y^k equal to x^i using only the features in *F'*. The inconsistency rate of a subset of features *F'* in the training set *T* is expressed as the average of the inconsistency scores of *T*'s instances with respect to *F'*. This is a monotonic measure, which has to be minimized.

4.2.3 Filter Methods

Filters perform feature selection independently of any particular classifier, being motivated by the properties of the data distribution itself. There are several robust algorithms in literature which employ a filter strategy. Among the most cited are: RELIEF [Kir92], LVF [Liu96], FOCUS [Alm97], Correlation-based filter – CFS [Hal00], or statistical methods based on hypothesis tests.

RELIEF [Kir92] is based on the idea employed by nearest neighbor learners: for each instance in a randomly chosen sample, it computes its nearest hit (closest instance from the same class) and miss (closest instance from a different class), and uses a weight update mechanism on the features. After all the training instances in the sample have been analyzed, the features are ranked according to their weights. The limitations of this method come from the fact that insufficient instances may fool it, and there is no general methodology for choosing the sample size.

LVF (Las Vegas Filter) [Liu96] uses a probabilistically-guided random search to explore the attribute subspace, and a consistency evaluation measure, different from the one employed by FOCUS. The method is efficient, and has the advantage of being able to find good subsets even for datasets with noise. Moreover, a good approximation of the final solution is available during the execution of the algorithm. One drawback is the fact that it may take longer to find the solution than algorithms using heuristic generation procedures, since it does not take advantage of prior knowledge.

FOCUS [Alm97] is one of the earliest multivariate filters. It is devised for binary class problems, and employs the min-features bias, meaning that it tries to find a minimal consistent feature set (a set which is able to separate the classes on the training data). Its drawbacks include the inability to handle noisy data and its predisposition towards over-fitting. Also, since it performs an exhaustive search, it is only tractable for small sets.

CFS (Correlation-based Feature Selection) [Hal00] is a filter method which selects those attributes which exhibit a strong correlation with the target attribute, and a weak correlation between each-other. For each candidate subset, a ratio of the group attribute-class correlation against attribute-attribute correlation is computed, as in equation (4.7). The subset which maximizes the ratio is the reduced attribute set.

PCA (Principal Components Analysis) is a filter method widely employed for feature selection and extraction in image processing applications [Ned09] (numeric attributes). It

performs an orthogonal transformation on the input space, to produce a lower dimensional space in which the main variations are maintained. There are several different versions to perform PCA – a review on several approaches is available in [Ned10].

4.2.4 Wrapper Methods

Since filters fail to capture the biases inherent in learning algorithms, for the purpose of boosting the classification performance, filter methods may not achieve significant improvements. Instead, wrapper methods should be considered. Experimental results which validate this assumption can be found in [Lan94a, Koh95]. Wrappers [Joh94, Koh95], as opposed to filter methods, search for the optimal subset by using an empirical risk estimate for a particular classifier (they perform empirical risk minimization). Thus, they are adjusted to the specific relations between the classification algorithm and the available training data. One drawback is that they tend to be rather slow.

In general wrapper methodology consists of three main steps:

- a generation procedure
- an evaluation procedure
- a validation procedure

Thus, a wrapper is a 3-tuple of the form <*generation, evaluation, validation*>. The feature selection process selects the minimal subset of features, considering the prediction performance as evaluation function: minimize the estimated error, or equivalently, maximize the expected accuracy. Each selected feature is considered to be (strongly) relevant, and rejected features are either irrelevant or redundant (with no further refinement). The differences in the application of the wrapper methodology are due to the methods used for generation, the classifier used for evaluation and the strategy for estimating the off-sample accuracy.

The *generation procedure* is a search procedure that selects a subset of features (F_i) from the original feature set (F), $F_i \subseteq F$, as presented in section 4.2.1. The *evaluation procedure* measures the quality of a subset obtained from a given generation procedure. As the selected feature subset depends on the evaluation function, the process of selecting the appropriate evaluation function is dependent on the particular initial dataset. In the case of wrappers the evaluation is performed by measuring the performance of a certain classifier on the projection of the initial dataset on the selected attributes (i.e. estimate the probability of error, as presented in section 4.2.2). The *validation procedure* tests the validity of selected subset through comparisons obtained from other feature selection and generation procedure pairs. The objective of the validation procedure is to identify the best performance that could be obtained in the first two steps of the method for a given dataset, i.e. to identify the selection method which is most suitable for the given dataset and classification method. As a consequence, the minimal feature subset is selected. All features from the subset are considered relevant to the target concept. Moreover, the classification method performs the best, so it is to be considered for further classifications.

The initial work on wrappers has been carried out by John, Kohavi and Pfleger [Joh94], which conducted a series of experiments to study the effect of feature selection on the generalization performance of ID3 and C4.5, using several artificial and natural domain datasets. The results indicated that, with one exception, feature selection did not change the generalization performance of the two algorithms significantly. Genetic search strategies were employed in [Che96] within a wrapper framework for decision tree learners (SET-Gen), in an attempt to improve both the accuracy and simplicity of the resulting models. The fitness function proposed by the authors was a linear combination of the accuracy, the size of the resulting trees (normalized by the training set size) and the number of features.

OBLIVION has been proposed in [Lan94b], to alleviate the effect of irrelevant features on the kNN classifier. The algorithm uses backward elimination as generation procedure and an oblivious decision tree as classifier [Koh95]. A context sensitive wrapper for instance based learners is proposed in [Dom97], which selects a (potentially) different subset of features for each instance in the training set. Backward elimination is employed as search strategy and cross-validation to estimate the accuracy. The method is especially useful in domains where the features present local relevance.

Improvements on the naïve Bayes classifier via the employment of wrapper-based feature selection are reported in [Paz95, Koh97].

RFE (Recursive Feature Elimination) [Guy02] is a combination of a wrapper and a feature ranking scheme. In each iteration, an SVM is trained on the current subset of features; then, a ranking of the features is computed from their weights in the model – i.e. the orientation of the hyperplane. The least important feature is removed and the process continues with the next iteration. The stopping criterion is typically a risk estimate (i.e. wrapper based), but the method can be used also to generate a ranking of the features.

The most important criticism brought to the wrapper approach is concerned with its computational cost, since each feature subset has to be evaluated by training and evaluating a classifier, possibly several times (if cross-validation, or repeated bootstrapping are used). To address this issue, efficient search strategies have been proposed in [Moo94] – race search and schemata search – and [Koh95] – compound search space operators. In addition, greedy search strategies have a reduced time complexity and seem to be robust against overfitting [Guy03].

4.3 Combining Generation Strategies

A first motivation for tackling a combination approach for the generation strategies can be found in the *no-free lunch theorem*. It is known that, due to the selective superiority of classifiers, there is no universally best method, i.e. one which yields superior performance to all other methods, on any problem. Intuitively, this issue should affect the generation strategies used in feature selection as well. As will be shown in section 4.3.2, there is no superior wrapper combination, although there are certain combinations which constantly yield good performance improvement. Different search strategies in the generation step may yield significantly different results.

A second motivation for such an approach is the fact that combination methods via ensemble learning or the Dempster-Shafer Theory of Evidence have been shown to improve the stability of individual classifiers across a wide range of problems [Mol07, Mur10]. Such approaches reduce the variance associated to single learners, and by combining different methods the resulting bias is expected to be lower than the average bias of the individual methods.

Also, wrappers are known to be significantly slower than filters, since they require training and evaluating a classifier for each attribute subset generated during the search process. Thus, using faster search strategies without affecting the quality of the solution is important.

As a result, this section proposes an original wrapper-based attribute selection method, which combines the selections of several generation procedures, via voting. The expected effect is an increased stability over several problems, while keeping a high reduction rate in the number of attributes.

The procedure is presented at the beginning of the next page. *T* is the available training set, *Sp* is the set of available generation procedures and *sEval* is the subset evaluation method, i.e. the strategy employed and the classifier used by the wrapper.

COMBINE GENERATION STRATEGIES

Given: Set $Sp = \{Sp_1, Sp_2, ..., Sp_p\}$ *of search strategies*
sEval – subset evaluation method
T – training set

Do:

1. *Generate individual feature subsets corresponding to each search method, using sEval and T:*

$$F_{CV} = \{F_{CV}^1, ..., F_{CV}^p\}, \text{ where } F_{CV}^k = \{(X_j, cv_j)^k \mid j = \overline{1,n}\}$$

cv_j^k *– local score of attribute* X_j *in set* F_{CV}^k

2. *Compute, for each attribute, a global score:*

$$s_j = \sum_{k=1}^{p} \omega_k cv_j^k$$

3. *Select the final attribute subset:*

$$F = \{X_j \mid s_j > \delta_j\}$$

where δ_j *is the selection threshold for attribute* X_j

F_{CV}^k is the feature subset generated by search strategy Sp_k individual, in which each feature possesses a corresponding selection score. One approach to generate such a feature subset is to run the wrapper in a cross-validation loop and assign to each feature a score equal to the number of folds in which it was selected. Using the local selection scores, we compute a global weighted score for each feature, and apply a selection strategy to obtain the final feature subset. Currently, an above average uniform selection strategy has been applied, but the method can be extended to accommodate other voting strategies.

4.4 Experimental Evaluation

4.4.1 Evaluating the Wrapper Methodology

A series of evaluations on the wrapper methodology have been conducted in order to study its capacity to improve the learning performance of classifiers. Accuracy has been employed as a measure of the classification performance. The possibility of combining different classifiers for the steps of feature selection and learning has been analyzed. It is known that the Bayesian classifier is able to deal with irrelevant features, but not with the redundant ones. On the contrary, decision trees exhibit good behavior in the presence of redundant features, but usually fail when dealing with irrelevant features. Evaluations have been performed to study the behavior of their combinations.

A second problem addressed by the evaluations is related to the use of pruning when wrapping feature selection around decision trees. The problem was originally formulated in [Koh95], where it was indicated that pruning should be avoided in this case.

Another issue analyzed in the current evaluations is related to the flow of the mining process. In [Mol07], it is argued that, due to the selective superiority of classifiers, the baseline accuracy of a dataset should be assessed before starting to mine a new real problem. A certain classifier is then considered appropriate for that problem only if it achieves a higher accuracy than the baseline accuracy. The question here is: does the feature selection step affect the initial selection of the learning scheme? Does the "most appropriate" algorithm for the given problem change after feature selection, or does it remain the same as in the initial choice?

In the attempt to provide answers to these questions, a series of comparative evaluations on several instantiations of the wrapper methodology have been performed. Fourteen UCI datasets were employed, described in table A.4.1 of Appendix A. In selecting the datasets, the criteria stated in [Koh95] were used: dataset size, reasonable encoding, comprehensibility, non-triviality and age. The evaluation scenarios have been set up using the WEKA framework [Wit05]. Following a series of preliminary evaluations on several search strategies, greedy stepwise backward search and best first search have been selected as generation procedures. Three different learning schemes, representing three prominent classes of algorithms: decision trees (C4.5 – revision 8 – J4.8, as implemented by WEKA); Naïve Bayes and ensemble methods (AdaBoost.M1, with Decision Stump as a base learner) were employed for the evaluation and validation procedures. For J4.8, experiments were performed both with and without pruning.

In presenting the results, the following abbreviations have been employed:

- for the generation procedure:
  - BFS: best first search
  - GBW: greedy stepwise backward search
- for the evaluation function and the validation procedure:
  - JP: J4.8 with pruning
  - JNP: J4.8 without pruning
  - NB: Naïve Bayes
  - AB: AdaBoost.M1

A "_" is used to signal a "don't care" situation (e.g. all combinations yield the same results).

Table 4.1 presents the results obtained by wrappers using the classifier which initially yielded the highest accuracy for the evaluation and validation procedures. In all but two cases we find that the accuracy increases after performing feature selection using the wrapper approach on the initially best classifier.

Table 4.1 – Results obtained by wrapper combinations using the initially best classifier

Dataset	**A1**	**M1**	**A2**	**M2**	**RI (%)**
Australian	86.26	JP	87.43	GBW/JP/JP	1.36
Breast-cancer	75.03	JP	75.67	_/JP/JP	0.85
Bupa	63.1	NB	65.17	BFS/NB/NB	3.28
Cleve_detrano	83.73	NB	85.74	BFS/NB/NB	2.40
Crx	84.93	JP	86.68	GBW/JNP/JNP	2.06
German	75.16	NB	75.72	BFS/NB/NB	0.75
Heart	83.13	NB	85.48	BFS/NB/NB	2.83
Cleveland	56.54	NB	60.71	GBW/NB/NB	7.38
Monk3	98.91	JP	98.92	_/J_/J_	0.01
PimaDiabethes	75.75	NB	77.58	BFS/NB/NB	2.42
Thyroid	99.45	AB	99.28	BFS/J_/J_	-0.17
Tic-tac-toe	83.43	JP	83.47	BFS/J_/JNP	0.05
Vote	96.22	JP	96.73	GBW/JP/JP	0.53
Wisconsin	96.24	NB	96.07	BFS/NB/NB	-0.18

A1 = Initial best accuracy; M1 = Initial best classifier; A2 = Accuracy for the wrapper method which uses the initial best classifier for both evaluation and validation; M2 = Wrapper method which uses the initial best classifier for both evaluation and validation; RI = Relative Improvement (%) = (A2-A1)/A1

On the Thyroid dataset no improvement is found. This behavior is explained by the very high initial accuracy. Due to that value, improvements are difficult to obtain. Thus, there is no reason for performing feature selection. The other exception is the Wisconsin dataset, for which NB attains the highest initial accuracy. Even if no improvement can be found when using this classifier, other combinations used for the wrapper lead to an accuracy increase on this dataset as well (see table 4.2).

Table 4.2 presents the results obtained by the best <generation, evaluation, validation> combinations. In most of the cases, the BFS/NB/NB combination achieves the highest accuracy, while combinations using JP or JNP come in second. There are three exceptions to this rule:

- the Breast-cancer dataset: the only dataset on which combinations employing AB obtain the highest accuracy
- the Cleveland dataset: here, GBW obtains the best accuracy. Also, the Cleveland dataset is the only one in which GBW selected fewer attributes than BFS.
- the Wisconsin dataset: as shown earlier, for this set a mix of classifiers in the evaluation/validation steps achieves the highest accuracy.

Table 4.3 presents the results obtained by the _/JNP/JNP wrapper. The great advantage of this wrapper is that it is extremely stable. It constantly boosts the accuracy, even though it does not obtain the best improvement all the time. The _/J_/J_ wrappers obtain the best accuracies on 4 datasets (out of 14), while in 1 case the best accuracy is obtained with a wrapper using J4.8 as evaluation function. Moreover, on 8 datasets, a wrapper based on J4.8 obtains the second best performance, while on 2 other datasets, it is considered as evaluation function, or validation function respectively. The best relative accuracy improvement of this wrapper is of 11% (on Heart dataset, for both search procedures), while the average improvements are 2.29% for BFS and 3.0% for GBW. Thus, instead of using J4.8 with pruning as the learning scheme, it is preferable to perform an initial feature selection using J4.8 without pruning, and then apply, in the learning step, again J4.8 without pruning.

Table 4.2 – Best wrapper combinations

Dataset	**A1**	**M1**	**A2**	**M2**	**RI (%)**
Australian	77.35	NB	87.58	BFS/NB/NB	13.23
Breast-cancer	72.38	AB	76.1	GBW/AB/AB	5.14
Bupa	63.1	NB	65.17	BFS/NB/NB	3.28
Cleve_detrano	83.73	NB	85.74	BFS/NB/NB	2.40
Crx	77.86	NB	87.51	BFS/NB/NB	12.39
German	75.16	NB	75.72	BFS/NB/NB	0.75
Heart	83.13	NB	85.48	BFS/NB/NB	2.83
Cleveland	56.54	NB	60.71	GBW/NB/NB	7.38
Monk3	98.91	JP	98.92	_/J_/J_	0.01
PimaDiabethes	75.75	NB	77.58	BFS/NB/NB	2.42
Thyroid	99.3	JP	99.28	N/A	-0.02
Tic-tac-toe	83.43	JP	83.47	BFS/J_/JNP	0.05
Vote	96.22	JP	96.73	GBW/JP/JP	0.53
Wisconsin	96.24	NB	96.48	BFS/JP/NB	0.25

A1 = Initial accuracy for the classifier used in the best wrapper; M1 = Classifier of the best wrapper; A2 = Accuracy for best wrapper; M2 = Best wrapper; RI = Relative Improvement

Table 4.3 – Results obtained by the _/JNP/JNP wrapper

Dataset	Initial	BFS	GBW	Dataset	Initial	BFS	GBW
Australian	86.2	86.26	86.03	*Cleveland*	53.46	53.32	59.55
Breast-cancer	73.68	75.03	75.19	*Monk3*	98.91	98.92	98.92
Bupa	59.13	64.80	64.80	*PimaDiabethes*	73.82	73.35	74.00
Cleve_detrano	76.63	82.08	80.33	*Thyroid*	99.3	97.73	97.78
Crx	84.93	86.14	86.68	*Tic-tac-toe*	83.43	83.47	83.47
German	71.72	73.11	72.26	*Vote*	96.22	96.64	96.66
Heart	76.16	84.85	84.85	*Wisconsin*	94.41	94.99	95.23

Table 4.4 shows how the number of attributes is significantly reduced through feature selection (54.65% and 52.33% on average, for the best wrapper, and respectively second best wrapper). This limits the search space and speeds up the training process. Generally, BFS selects fewer attributes than GBW, and the resulting datasets prove to be more efficient. The exception is the Cleveland dataset, for which GBW selects fewer attributes than BFS. In this case also the performance is better. The general conclusion is that fewer attributes lead to a better performance (both increased accuracy and reduced training time).

Table 4.5 shows the first and second best accuracies obtained after feature selection. It can be observed that the second best improvement is significant as well, which indicates that feature selection should be used in any data mining process, regardless of how good the available learning algorithm is.

4.4.2 Evaluating the Combination Strategy

A series of evaluations on 10 UCI benchmark datasets have been performed, to analyze the effects produced by the proposed combination method on the classification performance of J4.8. Four different search strategies have been considered: best-first search (bfs), bi-directional best-first search (bfs_bid), forward and backward greedy stepwise search (gsf and gsb, respectively). J4.8 has been employed for the wrapper evaluation function. 10-fold cross-validation has been used for both performance evaluation and in the execution of the combination method.

The accuracy values obtained by the various methods are presented in table 4.6, while the number of attributes selected by each method is displayed in table 4.7. As the results indicate, an individual method can achieve the best accuracy on a dataset and the worst on a different dataset, whereas the combination method always yields superior performance to the worst individual method.

Table 4.4 – Number of attributes selected

Dataset	N1	N2	N3	Dataset	N1	N2	N3
Australian	14	8	8	*Cleveland*	13	5	5
Breast-cancer	9	2	4	*Monk3*	7	3	3
Bupa	5	2	2	*PimaDiabethes*	8	5	5
Cleve_detrano	14	6	6	*Thyroid*	20	5	7
Crx	15	6	6	*Tic-tac-toe*	9	7	7
German	20	9	9	*Vote*	16	7	7
Heart	13	8	8	*Wisconsin*	9	5	5

N1 = Number of initial attributes in the dataset (without the class attribute); N2 = Number of attributes selected by the best wrapper method; N3 = Number of attributes selected by the wrapper which uses the best classifier (the classifier which achieved the best accuracy on the original dataset)

Table 4.5 – First and second best accuracies obtained after feature selection

Dataset	A1	A2	A3	RI2 (%)
Australian	86.26	87.58	87.43	1.36
Breast-cancer	75.03	76.10	75.67	0.85
Bupa	63.1	65.17	64.8	2.69
Cleve_detrano	83.73	85.74	84.88	1.37
Crx	84.93	87.51	86.68	2.06
German	75.16	75.72	75.56	0.53
Heart	83.13	85.48	85.33	2.65
Cleveland	56.54	60.71	60.49	6.99
Monk3	98.91	98.92	98.92	0.01
PimaDiabethes	75.75	77.58	77.06	1.73
Thyroid	99.45	99.28	99.27	-0.18
Tic-tac-toe	83.43	83.47	83.47	0.05
Vote	96.22	96.73	96.71	0.51
Wisconsin	96.24	96.48	96.32	0.08

A1 = initial accuracy for the best classifier; A2 = Accuracy for the best wrapper method; A3 = Accuracy for the second best wrapper method; RI2 = Relative improvement for the second best wrapper method

Also, its performance is similar to the average of the individual methods, and in several cases it achieves the best, or close to the best performance (6 out of 10 datasets). The Wilcoxon statistical signed ranked test has indicated that there is no significant statistical difference between the individual methods and the combination method (at p=0.05). Also, except for the GSF-based wrapper, there is a statistical difference between the performance of the individual methods and the initial performance, and also between the combination method and the initial performance. The stability of the selection is therefore achieved, thus reducing the risk of selecting an inappropriate method in a new problem.

The reduction in the number of attributes produced by the combination method is also significant – similar to the average reduction achieved by the individual generation strategies. The relative reduction to the initial attribute set is of ~62%.

Therefore, the combination method provides a correct assessment of the expected performance improvement via feature selection, by establishing a baseline performance level for the analyzed dataset and classification method.

Table 4.6 – J4.8 accuracies on attribute subsets resulted from wrapper subset selection with various search strategies

Dataset	Initial	BFS	BFS_bi	GSB	GSF	Average	Combination method
Breast-cancer	73.68	**75.67**	**75.67**	**75.67**	75.60	75.65	**75.67**
Cleve-detrano	76.63	79.84	78.86	78.64	77.28	78.66	**82.88**
Crx	84.93	85.87	85.36	**86.32**	85.49	85.76	**86.25**
German	71.72	73.82	74.12	73.85	**74.86**	74.16	73.88
Heart	76.16	**83.19**	82.00	80.19	**83.19**	82.14	**83.19**
Hepatitis	78.05	**83.59**	83.45	82.28	**83.59**	83.23	83.18
Labor	78.38	80.17	80.17	79.90	**81.63**	80.47	**81.63**
Lymphography	76.46	**82.90**	**82.90**	82.20	81.23	82.31	**82.90**
Pima diabethes	73.82	74.26	74.26	**75.73**	74.26	74.63	74.26
Tic-tac-toe	**83.43**	82.96	81.44	69.94	81.44	78.95	75.08

Table 4.7 – Size of attribute subsets resulted from wrapper subset selection with various search strategies

Dataset	Initial Attrib.	bfs Attrib.	bfs_bid Attrib.	gsb Attrib.	gsf Attrib.	Average Attrib.	Combination Attrib.
Breast-cancer	9	4	4	4	**3**	4	4
Cleve-detrano	13	7	5	5	5	6	**5**
Crx	15	5	6	6	4	5	8
German	20	10	7	10	8	9	9
Heart	13	4	5	7	4	5	**4**
Hepatitis	19	3	4	10	3	5	7
Labor	17	6	6	7	4	6	**4**
Lymphography	18	6	6	8	4	6	6
Pima diabethes	8	3	3	3	3	3	**3**
Tic-tac-toe	9	7	6	6	1	5	3

4.5 Conclusions on Feature Selection

Among the many possible advantages of feature selection, perhaps the most important is improving the classification performance. All feature selection methods can be modeled as a combination of three steps: generation, evaluation and validation. The different alternatives available for achieving each step provide for a very wide spectrum of feature selection methods. However, just like in the case of learning algorithms, there is no universally best feature selection method. For the purpose of performance improvement, wrappers provide the most appropriate strategy. An original contribution presented in this dissertation is the systematic analysis and the identification of the most promising combinations of search methods for generation, and classifiers for evaluation and validation, such that the performance (i.e. the accuracy) increase is guaranteed.

As the experimental results prove, wrappers can always improve the performance of classifiers. In most cases, the classifier which initially achieved the highest accuracy maintains its high performance after feature selection (first or second best performance). This means that once a dataset has been initially assessed and a certain learning scheme has been selected as being appropriate, that scheme will maintain its performance throughout the mining process. Also, for all the datasets considered, the second best performance after feature selection still yields significant improvements over the initial classifier, which proves the necessity for such a step.

Although there is no absolute best method, BFS/B/B achieves the highest accuracy in most of the cases. The wrapper _/JNP/JNP achieves the most significant improvements relative to the initial accuracy (up to 11%). The number of attributes is considerably reduced (over 50%), which results in faster training, yet another advantage of attribute selection.

In the attempt to reduce the bias introduced by the search methods used in the generation procedure and improve the stability of feature selection, without increasing its complexity, an original combination method has been proposed, which selects the most appropriate attributes by applying a global selection strategy on the attribute subsets selected individually by the search methods. Greedy methods have been considered for combination, since they provide good quality solutions relatively fast. The evaluations performed on the newly proposed method have confirmed that the method achieves better stability than individual feature selection performed via different search methods, while keeping the high reduction level. The method can be employed for initial problem assessment, to establish a baseline performance for feature selection.

The original combination method and the analysis presented in this chapter have been acknowledged by the research community through the publication of 2 research papers in the proceedings of renowned international conferences:

1. **Vidrighin**, **B.C.**, Muresan, T., Potolea, R.,"Improving Classification Accuracy through Feature Selection", *Proceedings of the 4th IEEE International Conference on Intelligent Computer Communication and Processing*, pp. 25-32, 2008
2. **Vidrighin, B.C.**, Potolea, R., "Towards a Combined Approach to Feature Selection", *Proceedings of the 3rd International Conference on Software and Data Technologies*, pp. 134-139, 2008

5 Joining Pre-processing Steps: A Methodology

Even if significant efforts have been conducted to develop methods which handle incomplete data or perform feature selection, with notable achievements in both fields independently, to our knowledge there hasn't been any attempt to address the two issues in a combined manner. This is what is proposed in this chapter – a joint feature selection – data imputation pre-processing methodology.

5.1 A Joint Feature Selection – Data Imputation Methodology

The novelty of our methodology consists in the enhancement of the data imputation step with information provided by the attribute selection step. It considers the pre-processing activity as a homogeneous task, joining the two formerly independent steps:

- attribute selection
- data imputation

More specifically, the methodology explicitly performs attribute selection for the data imputation phase, i.e. only the values of the attributes which are relevant for the attribute being imputed are employed when determining the replacement value. The methodology imposes neither the technique for attribute selection, nor the data imputation technique. However, an imputation technique based on supervised learning methods should be employed, in order to make use of the selected attributes.

There are two variants of the methodology. One performs data imputation first and then attribute subset selection, and the second which uses the reverse order for the two. Subsequently, we employ the following abbreviations:

- F – the original attribute set in the training set
- CT – subset of complete training instances
- IT_j – subset of training instances with value for X_j missing
- AOS_j – the predictive attribute subset for X_j
- COS – the predictive attribute subset for the class Y

5.1.1 FSAfterI

In the *FSAfterI* version of the methodology, each attribute X_j except the class is considered. If there are any instances in the training set with unknown values for the current attribute, the methodology considers the attribute for imputation. CT represents a subset of complete instances. For each attribute X_j in turn, a subset containing the incomplete instances with respect to the attribute is extracted. The complete subset, CT, is used to build the imputation model for the incomplete part, as described further: the feature subset predictive for attribute X_j, AOS_j, is extracted from the complete training subset. Then, a model is built for attribute X_j using the complete instances subset and the features in AOS_j. Using this model, the replacement values for the incomplete instances are computed and imputed in the initial training set. Thus, the training set becomes complete in X_j. After all the attributes have been considered, all instances in the training set are complete. At this point, feature selection is applied on the training set to determine the class-predictive feature subset, COS. The projection of the training set on COS is the result of the procedure.

Performing data imputation first may induce a bias in the attribute selection step for determining the class-optimal attribute subset. Thus, an irrelevant attribute could be selected in COS due to its imputed values.

```
FSAfterI

CT    = extract_complete(T)
T'    = T
For each attribute Xj
      AOSj  = fSelect(F – {Xj}∪{Y}, CT, Xj)
      T'    = Impute(prAOSjT', Xj)
COS = fSelect(F,T',Y)
Tresult = prCOST'
```

5.1.2 FSBeforeI

The second version of the proposed methodology, *FSBeforeI*, considers the two phases in reverse order. It does not include any bias in the class-optimal attribute selection phase, since the operation is performed before the imputation phase. For determining AOS_j the original feature set F and the class Y are employed. Thus we ensure that all the relevant attributes for X_j are employed to build the imputation model. For performing feature selection, in both the generation of *COS* and AOS_j, k-fold cross-validation is employed, and the attributes which are "better" on the average are selected. The tactic for quantifying "better" on the average depends on the feature selection method employed: ranking methods yield average merit/rank measures, while other methods may indicate a percentage corresponding to the number of folds in which the attribute has been selected. Based on this information, the predictive subset can be deduced.

```
FSBeforeI

CT    = extract_complete(T)
COS = fSelect(F,CT,Y)

T'    = T
For each attribute Xj in COS
      AOSj  = fSelect(F – {Xj}∪{Y}, CT, Xj)
      T'    = Impute(prAOSjT', Xj)
Tresult = prCOST'
```

5.2 Experimental Evaluation

We have performed several evaluations with different implementations of the combined methodology, implemented within the WEKA framework. Initial experiments have been conducted on 14 benchmark datasets, obtained from the UCI repository. The current evaluations have been conducted on the following complete UCI datasets: Bupa Liver Disorders, Cleveland Heart Disease, and Pima Indian Diabetes (described in Appendix A, table A.5.1).

The following specializations of the methodology have been considered:

- For attribute selection (f):
  - ReliefF [Kon94]
  - CFS – Correlation-based Feature Selection [Hal00]
  - Wrapper
- For data imputation (i):
  - kNN – k Nearest Neighbor (denoted also as IBk)

- For evaluating the performance (c): the average classification accuracy computed using 10 trials of a stratified 10-fold cross validation for:
    - J4.8
    - Naïve Bayes (NB)

The search method (generation procedure) employed in the attribute selection is best-first search. The predictive attribute subset is obtained via 10-fold cross validation. For imputation with kNN, k has been set to 5. We have employed the following evaluation strategy:

- Incompleteness has been simulated, at different levels, using the strategy described in section 3.3.2.
- In each trial of a stratified 10-fold cross-validation, for each attribute A_i in the trial training set (except the class) vary the percentage of incompleteness. Then apply the pre-processing methodology, in its current specialization, to obtain the preprocessed trial training set. Finally, build a model from the modified training set, using a classification algorithm, and estimate its accuracy using the trial testing set.
- In addition, for each attribute A_i, the average classification accuracy has been estimated for different versions of the training set: complete, incomplete, imputed and pre-processed through feature selection.

The evaluations attempt to validate empirically the following statements: the combination is more efficient than the individual steps it combines, and the specializations of the combination are stable across the attributes of a dataset – the same specialization is identified as being the best for all the significant attributes. By significant attribute we mean the attributes which are constantly selected by different feature selection techniques. These are the attributes that will influence the most the quality of the learned model.

The diagrams (a) – (d) from figure 5.1 present the results obtained by the *FSAfterI* specialization of the methodology on the significant attributes of the Pima dataset, as computed in Chapter 3: attribute 2 – Glucose test and attribute 6 – Body-Mass Index. Each curve in the diagrams represents the accuracy obtained by the given specialization of the methodology. The performance of the classifiers on the complete and p% incomplete datasets have also been recorded. For both attributes considered, the most stable improvements are obtained by specializations for NB (~1% absolute improvement). For attribute Body-Mass Index, the Wrapper specialization yields good results for J4.8 as well (up to 4%).

For the significant attributes of the Bupa dataset, considerable improvements have been obtained by specializations for J4.8 (1-2% absolute improvement achieved by the CFS specializations, 1-3% by the wrapper up to 1% by the ReliefF based specializations). However, NB seems to be more efficient on the incomplete dataset, and the pre-processing methodology cannot, generally, boost J4.8's accuracy over NB's level (with the exception of the wrapper based specialization). Therefore, specializations for NB should be considered here as well.

For the Cleveland dataset, the most significant improvements are obtained by specializations for J4.8 (3-4% achieved by the CFS based specialization). However, the ReliefF specializations for NB yields the highest accuracy levels: ~58% for both attributes analyzed, as opposed to ~56% - the accuracy obtained by NB on the incomplete versions of the dataset, and ~56-56.5% - the accuracy obtained by the best specialization for J4.8 (using CFS). The results presented so far have indicated that we can perform the selection of the learning algorithm prior to performing pre-processing with the proposed methodology.

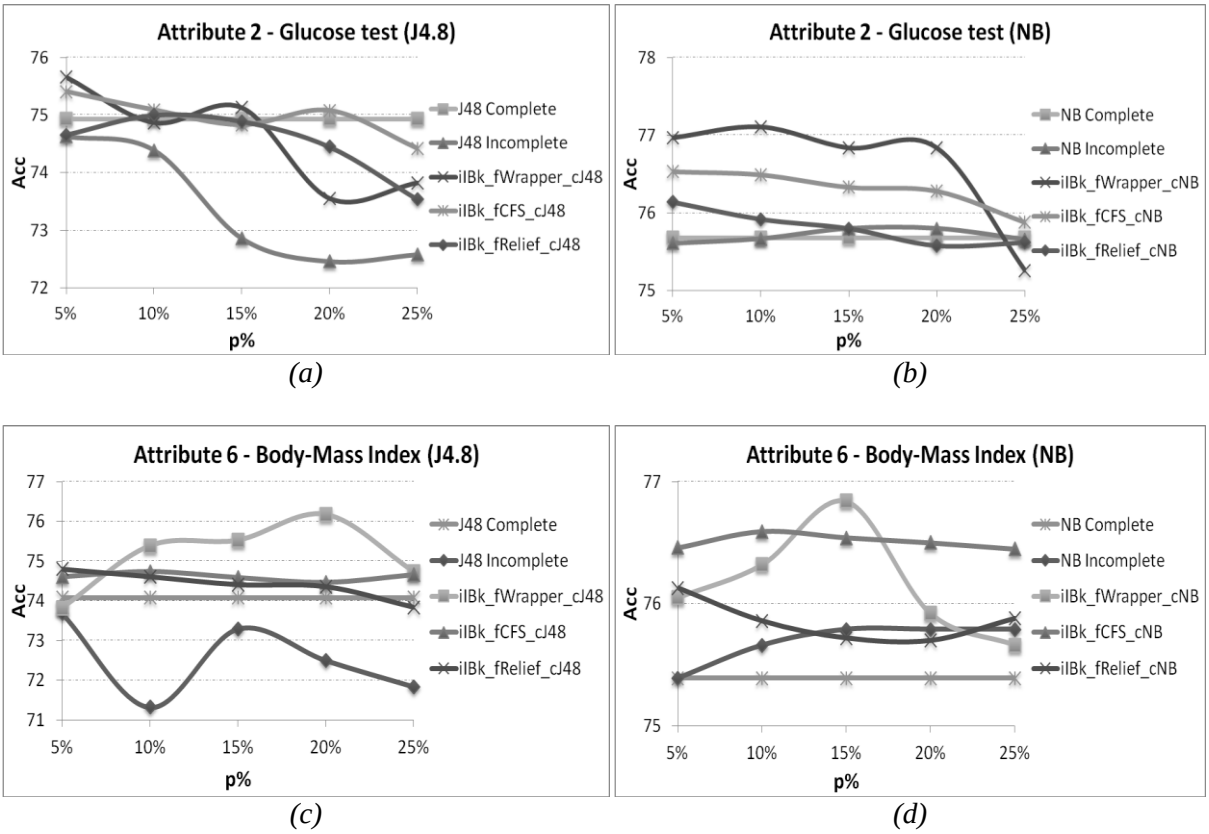


Figure 5.1– Accuracy obtained by different FSAfterI specializations, when compared to the accuracy on the incomplete dataset, for attributes strongly correlated with the class, Pima dataset

In the following, we present a comparative analysis for the classification performance on different versions of the training set, obtained through several pre-processing strategies: imputation, attribute selection and the combined methodology (both *FSAfterI* and *FSBeforeI*). Also, the classification performance on the complete and p% incomplete datasets is reported. Tables 5.1-5.2 present the accuracy levels obtained for two significant attributes of the Cleveland dataset. The specialization considered for the combined methodology employs CFS for attribute selection. In both cases, the two versions of the combined methodology yield better classification accuracies than the incomplete dataset (up to 5% absolute increase). A clear improvement can be observed over the imputation step also (up to 5% absolute increase, the rows in dark grey shading in the tables). The performance of feature selection approaches is similar to that of our proposed methodology on this dataset.

5.3 Conclusions

There exist a series of pre-processing tasks and associated techniques which focus on preparing the raw data for the mining step. However, each technique focuses on a single aspect of the data, and there is no information exchange between independent pre-processing steps. This chapter presents a new approach for pre-processing, which joins two formerly independent pre-processing steps: data imputation and feature selection. The methodology explicitly performs attribute selection for the data imputation phase.

Two formal versions of the proposed methodology have been introduced: *FSAfterI* and *FSBeforeI*. The two differ in the order of the two phases of the methodology: the first performs data imputation first and then selects the class-optimal feature subset, while the second considers the reverse order. *FSBeforeI* should be preferred, since it doesn't introduce any imputation bias in the feature selection phase.

Several particularizations of the methodology have been implemented, using two different filter attribute selection techniques and a wrapper, two classification methods and an imputation method. The resulting particularizations have been evaluated comparatively on benchmark data, using the accuracy of the same classification algorithms on the incomplete versions of the datasets as reference performance. The results have shown that the joint pre-processing methodology generally improves the performance of the classification algorithm, when using the preprocessed training set, as compared to the performance it obtains on the incomplete training set. Although there is no single definite winner combination for all datasets, a best combination can be usually identified for a particular dataset. Moreover, specializations using CFS for attribute selection and NB for the final classification have always yielded higher accuracy levels when compared to the accuracy on the incomplete data. Also, the combination proves to be superior to the data imputation and attribute selection tasks performed individually, which recommends it as a robust approach for performing data pre-processing.

The results have indicated that, in most cases, the improvement over the imputation task is significant (an absolute increase in accuracy of up to 5%). As for the comparison with the individual attribute selection task, in most situations the performance of the combined methodology is superior to that of the attribute selection step (absolute improvement of up to 1%). In the exception cases, feature selection yields the highest performance of all the other approaches.

The original data pre-processing methodology proposed in this chapter is the result of research supported by PNII grant no.12080/2008: SEArCH – Adaptive E-Learning Systems using Concept Maps. The proposed method has been accepted by the research community through the publication of two research papers in the proceedings of renowned international conferences:

1. **Vidrighin, B.C.,** Potolea, R., "Unified Strategy for Feature Selection and Data Imputation", *Proceedings of the 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, Timisoara, 26-29 sept. 2009, pp. 413 – 419
2. **Vidrighin, B.C.,** Potolea, R.,"Towards a Unified Strategy for the Preprocessing Step in Data Mining", *Proceedings of the 11th International Conference on Enterprise Information Systems*, pp. 230-235, 2009

Table 5.1 – The average accuracy (and standard deviation) obtained by J4.8 on different versions of the training set and different incompleteness levels (5-30%), for attribute STDepression, Cleveland dataset (specialization iIBk_fCfsSubsetEval_cJ48)

ATTRIBUTE STDepression	***5%***		***10%***		***15%***		***20%***		***25%***		***30%***	
	Acc.	***Stdd.***	***Acc.***	***Stdd.***	***Acc.***	***Stdd.***	***Acc.***	***Stdd.***	***Acc.***	***Stdd.***	***Acc.***	***Stdd.***
COMP (Complete)	52.38 (2.6)											
INC(Missing)	53.41	2.64	52.31	2.65	52.83	3.06	52.66	2.64	52.83	2.64	52.79	2.64
IMP (Imputation)	52.86	2.12	53.17	2.17	53.62	2.68	54.07	2.59	53.07	2.4	53.55	2.64
FS(Feature Selection)	57.03	1.95	**57.66**	1.85	57.07	1.83	**57.14**	1.83	57.55	1.83	57.83	1.83
FSAfterI	57.07	1.11	57.31	1.37	56.97	1.55	56.45	1.17	56.79	2.77	57.17	1.97
FSBeforeI	**57.1**	1.01	57.17	1.1	**57.34**	1.72	56.31	1.76	**57.9**	2.67	**58**	2.04
COMPL-IMP	-0.48	-	-0.79	-	-1.24	-	-1.69	-	-0.69	-	-1.17	-
IMP-INC	-0.55	-	0.86	-	0.79	-	1.41	-	0.24	-	0.76	-
FSAfterI –INC	3.66	-	5	-	4.14	-	3.79	-	3.97	-	4.38	-
FSAfterI –IMP	4.21	-	4.14	-	3.34	-	2.38	-	3.72	-	3.62	-
FSAfterI –FS	0.03	-	-0.34	-	-0.1	-	-0.69	-	-0.76	-	-0.66	-
FSAfterI –COMP	4.69	-	4.93	-	4.59	-	4.07	-	4.41	-	4.79	-
FSBeforeI –INC	3.69	-	4.86	-	4.52	-	3.66	-	**5.07**	-	5.21	-
FSBeforeI –IMP	4.24	-	4	-	3.72	-	2.24	-	4.83	-	4.45	-
FSBeforeI –FS	0.07	-	-0.48	-	0.28	-	-0.83	-	0.34	-	0.17	-
FSBeforeI –COMP	4.72	-	4.79	-	4.97	-	3.93	-	5.52	-	5.62	-

Table 5.2 – The average accuracy (and standard deviation) obtained by J4.8 on different versions of the training set and different incompleteness levels (5-30%), for attribute Thal, Cleveland dataset (specialization iIBk_fCfsSubsetEval_cJ48)

ATTRIBUTE Thal	***5%***		***10%***		***15%***		***20%***		***25%***		***30%***	
	Acc.	***Stdd.***	***Acc.***	***Stdd.***	***Acc.***	***Stdd.***	***Acc.***	***Stdd.***	***Acc.***	***Stdd.***	***Acc.***	***Stdd.***
COMP (Complete)	52.38 (2.6)											
INC(Missing)	52.79	2.43	53.28	1.97	53.41	2.77	54.03	2.7	53.66	2.94	53.97	2.58
IMP (Imputation)	52.66	1.62	51.93	1.74	53.21	2.46	52.76	2.95	53.14	2.53	53.48	3.2
FS(Feature Selection)	56.79	1.48	56.55	1.88	55.76	1.9	56.21	1.87	56.45	2.81	**56.55**	2.84
FSAfterI	57.69	1.05	56.72	2.24	**56.66**	1.26	57	1.4	**57.31**	1.77	56.38	1.48
FSBeforeI	**57.76**	1.34	**56.83**	2.08	56.41	1.76	**57.07**	1.7	57.1	1.68	56.38	2.25
COMPL-IMP	-0.28	-	0.45	-	-0.83	-	-0.38	-	-0.76	-	-1.1	-
IMP-INC	-0.14	-	-1.34	-	-0.21	-	-1.28	-	-0.52	-	-0.48	-
FSAfterI –INC	4.9	-	3.45	-	3.24	-	2.97	-	3.66	-	2.41	-
FSAfterI –IMP	5.03	-	4.79	-	3.45	-	4.24	-	4.17	-	2.9	-
FSAfterI –FS	0.9	-	0.17	-	0.9	-	0.79	-	0.86	-	-0.17	-
FSAfterI –COMP	5.31	-	4.34	-	4.28	-	4.62	-	4.93	-	4	-
FSBeforeI –INC	4.97	-	3.55	-	3	-	3.03	-	3.45	-	2.41	-
FSBeforeI –IMP	**5.1**	-	4.9	-	3.21	-	4.31	-	3.97	-	2.9	-
FSBeforeI –FS	**0.97**	-	0.28	-	0.66	-	0.86	-	0.66	-	-0.17	-
FSBeforeI-COMP	5.38	-	4.45	-	4.03	-	4.69	-	4.72	-	4	-

6 Classification in Practice I: Imbalanced Error Costs and Expensive Tests

6.1 Problem Statement

The traditional approach to classification considers the error reduction strategy, i.e. the cost function uniformly quantifies the difference between the actual and the predicted class values, regardless of the class. Therefore, all errors are equally important. In real-world domains, however, there are numerous scenarios in which non-equal misclassification costs are inherent [Faw97, Pen03]. In fraud detection, for example, false negatives are clearly more harmful than false positives. The same situation occurs in medical diagnosis, where failing to identify a positive is almost unacceptable, whereas a certain level of false positive predictions is manageable. In contextual advertising, on the other hand, posting an irrelevant ad for the current context should be penalized more than failing to identify all the relevant ads. This is because only the top two or three ads are considered at a time and therefore it is more important that those predicted as being relevant are actually relevant, than to identify as many relevant ads as possible.

Moreover, there are situations in which the effort of acquiring certain data items has to be considered. Turning again to the medical diagnosis problem, another particularity is the fact that medical tests are usually costly (economically). Moreover, in terms of patient comfort, they normally range from mildly uncomfortable to painful. Also, collecting test results may be time-consuming. Arguably, time and patient comfort may not be real costs, but they do have some implication for the decision on whether it is practical to take a certain test or not. Performing all possible tests in advance is unfeasible and only a relevant subset should be selected. The decision on performing or not a certain test should be based on the relation between its cost and potential benefits. When the cost of a test exceeds the penalty for a misclassification, further testing is no longer justified.

These particularities are addressed by *cost-sensitive learning*, which is directed towards the reduction of the total cost, instead of just minimizing the number of misclassification errors. [Tur00] provides taxonomy of all the types of costs involved in inductive concept learning, the most important being the *misclassification costs* and the *test costs*. The situations which they cover are the ones detailed in the two paragraphs above: misclassification costs attempt to capture the non-uniform gravity of errors, while test costs quantify various aspects related to the acquisition of certain data values – monetary cost, acquisition time, pain inflicted, a.s.o.

The misclassification costs are represented via a cost matrix $M = (c_{ij})_{nxn}$, where c_{ij} represents the cost of misclassifying an instance of class j as being of class i. For binary classification problems, $n=2$:

$$M = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} \tag{6.1}$$

The main diagonal elements (c_{11} and c_{22}) represent the costs of correct identification and are normally smaller than or equal to 0 (i.e. reward or no penalty); c_{12} is the cost of a false positive (i.e. failing to identify a negative), while c_{21} captures the reverse situation. One of the most important difficulties when dealing with different error costs is quantifying misclassification costs. Even if it is relatively easy to determine which errors are more severe than others (e.g. in medical diagnosis $c_{12} > c_{21}$), it is difficult to quantify the gravity of an error, since this may translate, indirectly, into more serious social/moral dilemmas, such a putting a price tag on human life.

Test costs are associated with the predictor attributes, i.e. each attribute has an associated scalar cost value, which represents the overall cost, including monetary, time or

pain. Even if it is straightforward to quantify each of the categories involved, combining them into a single value may be difficult. Moreover, in applications in which both types of costs be considered, such as medical diagnosis, correct calibration of the misclassification costs to the test costs is essential.

6.2 State of the Art in Cost-Sensitive Learning

This section reviews the main strategies for cost-sensitive learning existing in literature. Mainly, there exist two categories of approaches: those which focus on minimizing misclassification costs and those which focus on test costs. Significantly less effort has been invested in combining the two strategies.

6.2.1 Reducing Misclassification Costs

Misclassification costs are considered to be the most important in cost-sensitive learning, and, therefore, the literature contains a rich spectrum of techniques and strategies for handling this type of costs. Roughly, the approaches can be grouped into: *direct methods*, which include adaptations to existing algorithms or newly developed techniques and *indirect (meta-) methods*, which refer to general approaches, applicable to any base classifier – such as sampling, or meta-learning.

Direct methods

This category of methods attempts to introduce and utilize misclassification costs directly into the learning algorithms. Most such approaches target modifications to the training and/or prediction stage(s) of decision trees, so as to make them cost-sensitive. One of the earliest such methods is *I-gain* [Paz94], which considers the misclassification costs at prediction time, i.e. instead of predicting the most probable class it predicts the class having the least expected cost. *GINI Altered Priors* [Paz94] applies Breiman's altered prior probabilities principle, which takes costs into consideration by weighting the prior probabilities of each class with the relative cost of misclassifying the examples of that class. [Paz94] incorporates this correction during training, with the GINI index attribute selection scheme, but the strategy can be employed with any entropy-based selection scheme. [Tin98, Tin02] proposes a strategy of incorporating misclassification costs via weights, similar to *stratification* [Mar03]. In this approach, a weight is assigned to each example, to reflect its importance – i.e. use costs as example weights, during the tree training and pruning processes. The *Linear Machines Decision Tree (LMDT)* method [Dra94] builds a multi-variate decision tree, by training a linear machine at each internal node and having a class label at each leaf. Misclassification costs are considered during the training of the linear machines – in the weight learning and feature elimination processes.

Several strategies have been proposed for modifying the basic Support Vector Machines (SVMs) learning algorithm to achieve cost-sensitivity: (1) the *boundary movement* method (*BM-SVM*) [Kar98] shifts the decision boundary of the classical SVM by changing the threshold; (2) the *biased penalties* method *(BP-SVM)* [Bac06] introduces different penalty factors for the SVM slack variables during training; (3) [Mas10] propose a cost-sensitive loss function for the standard SVM algorithm and prove that the new cost-sensitive SVM achieves cost sensitivity for both separable and non-separable training data, enforcing a larger margin for the preferred class. Evaluations have yielded superior performance to methods (1) and (2).

Indirect methods

Perhaps one of the most straightforward indirect solutions for the problem of reducing the total misclassification cost was a procedure called *stratification* [Mar03]. In this approach, the actual classifier is not altered in any way; instead, the distribution of examples for each class is changed. The modified training set includes proportionally more examples of

the classes having high misclassification costs and may be generated either by under-sampling or by over-sampling. Each alternative comes at a certain price ([Dom99] contains a detailed discussion on the subject), but the most serious limitation of the approach is that it restricts the dimension or the form of the misclassification cost matrix – the technique is only applicable to two-class problems or to problems where the cost is independent of the predicted class. Costing [Zad03] is a meta-method which weighs the training instances according to the costs. It can be achieved either by feeding the weights directly to the classification algorithm (if possible), or by guided sub-sampling.

More complex techniques, which overcome the limitations of sampling, involve meta-learning algorithms, which typically are applicable to a range of base classifiers. In this category we include algorithms based on various *ensemble* methods, the most important being *boosting* [Fre97]. It works by combining several learners through voting; the resulting composite classifier generally has a higher predictive accuracy than any of its components. Each distinct model is build through the same learning mechanism, by varying the distribution of examples in the training set. After each boosting phase, the weights of the misclassified examples are increased, while those for the correctly classified examples are decreased. It has been mathematically proved that the error rate for the composite classifier on the un-weighted training examples approaches zero exponentially with an increasing number of boosting steps [Fre97, Qui96]. Also, various experimental results report that the reduction in error is maintained for unseen examples. The base classifiers may be either weak learners or more elaborate, cost-sensitive learners. The most prominent algorithms which apply this strategy are *AdaBoost.M1* [Fre97] and *AdaCost* [Fan00].

Another solution for reducing misclassification costs is *MetaCost* [Dom99]. The algorithm is based on the Bayes optimal prediction principle, which minimizes the conditional risk of predicting that an example belongs to class *i*, given its attributes *x*. The solution requires accurate estimates for the class probabilities of the examples in the training set. This distribution is obtained through an ensemble method, by uniform voting from individual classifiers. Once the conditional probabilities are estimated, the algorithm re-labels the examples in the training set, according to their optimal predictions and generates the final classifier, using the modified training set. The main advantages of this procedure are related to its applicability to wide range of base classifiers, the fact that it generates a single, understandable model, and its efficiency under changing costs (the conditional probabilities need to be computed only once, after which they can be used to generate models for various cost matrices).

Cost-Sensitive Classifier (CSC) [Wit05] is a meta-approach which makes error-reduction classifiers cost-sensitive. Two alternatives may be employed to induce cost-sensitivity: *(1)* reweight training instances according to the total cost assigned to each class (similar to costing) or *(2)* predict the class with minimum expected misclassification cost (rather than the most likely class) – similar to I-gain.

Empirical Thresholding (ET) [She06] is another meta-approach which can transform an error-reduction classifier into a cost-sensitive one, by selecting a proper threshold from training instances according to the misclassification cost; experiments conducted by the authors have indicated that the method has the least sensitivity on the misclassification cost ratio.

6.2.2 Reducing Test Costs

Several approaches exist also for tackling the problem of test costs. They are direct methods based on the decision tree paradigm, and typically involve some alteration of the information gain function, as to make it cost-sensitive. Various cost dependent functions have been proposed in the literature. Perhaps the best known algorithm in this category is *Eg2*

[Nun91] which uses the Information Cost Function (ICF) as the attribute selection criterion. For the i^{th} attribute, ICF may be defined as follows:

$$ICF_i = \frac{2^{\Delta I_i} - 1}{(C_i + 1)^w} \tag{6.2}$$

where $0 \leq w \leq 1$.

This means that the attribute selection criterion is no longer based solely on the attribute's contribution to obtaining a pure split, but also on its cost, C_i. Also, the Information Value Function contains parameter *w*, which adjusts the strength of the bias towards lower cost attributes. Thus, when $w = 0$, the cost of the attribute is ignored, and selection by ICF is equivalent to selection by the information gain function. On the other hand, when $w = 1$, ICF is strongly biased by the cost component.

IDX [Nor89] is a top-down inductive decision tree learner that chooses the attribute which maximizes the following heuristic:

$$\frac{\Delta I_i}{C_i} \tag{6.3}$$

where ΔI_i represents the information gain of attribute *i*, and C_i is is its cost.

CS-ID3 [Tan89] uses a lazy evaluation strategy, constructing only part of the tree that classifies the current case. Its attribute selection function is very similar to that of *IDX*, with the difference that the information gain of an attribute carries a heavier weight:

$$\frac{(\Delta I)^2}{C_i} \tag{6.4}$$

6.2.3 Reducing the Overall Cost

Significantly less work has been done for aggregating several cost components. The most prominent approach in the literature is *Inexpensive Classification with Expensive Tests – ICET* [Tur95], which combines a greedy search heuristic (decision tree) with a genetic search algorithm. A detailed discussion on ICET is provided in section 6.3ss.

Other, less known approaches generally consider test and misclassification costs as the attribute selection process of decision trees, instead of using one of the classical entropy-based principles [Paz94, Lin04]. *Cost-Sensitive Naïve Bayes (CSNB)* [Cha04] propose the employment of a test strategy to determine how unknown attributes are selected in order to minimize the sum of the misclassification and test costs.

6.3 ProICET: Enhancements on a Cost-sensitive Classifier

6.3.1 ICET: Inexpensive Classification with Expensive Tests

Classical tree induction uses *hill climbing* search, which, as most greedy techniques, suffers from the *horizon effect*, i.e. it tends to get caught in local optima. One possible way to avoid the pitfalls of simple greedy induction is to perform a heuristic search in the space of possible decision trees through evolutionary mechanisms. *ICET* (*Inexpensive Classification with Expensive Costs*) is such a hybrid algorithm. Introduced by Peter Turney, the technique tackles the problem of cost-sensitive classification by combining a greedy search heuristic (decision tree) with a genetic algorithm [Tur95].

The ICET algorithm has the following key features:

- It is sensitive to test costs;
- It is sensitive to misclassification costs;
- It combines a greedy search heuristic with a genetic search algorithm

These key features make ICET a very promising candidate for solving problems like medical diagnosis and prognosis, credit risk assessment or oil-slick detection, where different

errors cost different amounts, and where making a classification is costly in that the tests that have to be taken to determine attribute values carry a certain cost.

As described in [Tur95], the ICET algorithm can be viewed as working on two levels:

- On the bottom level, a greedy search in the space of decision trees is performed
- On the top level, the evolutionary component performs a genetic search through a space of biases; the biases are used to control EG2's preference for certain types of decision trees.

The algorithm starts by the genetic component evolving a population of randomly generated individuals (an individual corresponds to a decision tree). Each individual in the initial population is then evaluated by measuring its fitness. Standard mutation and crossover operators are applied to the trees population and, after a fixed number of iterations, the fittest individual is returned. The genetic algorithm used in the original work is GENESIS [Gre86], and the decision tree algorithm is a modified version of Quinlan's C4.5 [Qui93] which uses ICF (Information Cost Function) as attribute selection function, same as in EG2. An important remark is that, unlike EG2, ICET does not minimize test costs directly. Instead, it uses ICF for the codification of the individuals in the population. The *n* costs, C_i, are not true costs, but *bias parameters*. They provide enough variation to prevent the decision tree learner from getting trapped in a local optimum, by overrating/underrating the cost of certain tests based on past trials' performance. However, it is possible to use true costs, when generating the initial population, which has been shown to lead to some increase in performance [Tur95].

Each individual is represented as a bit string of *n + 2* numbers, encoded in Gray. The first *n* numbers represent the bias parameters ("alleged" test costs in the ICF function). The last two stand for the algorithm's parameters *CF* and *w*; the first controls the level of pruning (as defined for C4.5), while *w* is needed by ICF.

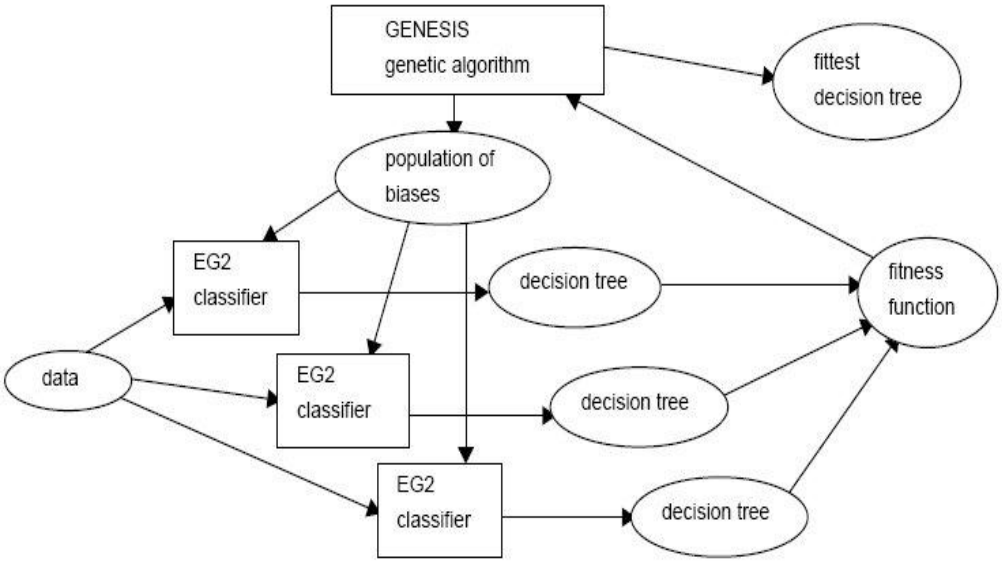


Figure 6.1 – ICET algorithm flow2

Each trial on an individual consists in training and evaluating a decision tree on a given dataset, using the biases in the individual to set the attribute costs, *CF* and *w*. This is done by splitting the available dataset into two subsets: sub-training and sub-testing dataset.

[2] [Tur95]

Since the split is random, there may be that two identical individuals will yield different outcomes (since the form of a decision tree is strongly related to the distribution in the training set – different training sets produce different trees). The evaluation of the tree gives the fitness function needed by the genetic algorithm to evaluate the individuals.

In ICET, the *fitness function* for an individual is computed as the *average cost* of classification of the corresponding tree obtained by randomly dividing the training set in two subsets, the first used for the actual tree induction and the second for error estimation). The average cost of classification is obtained by normalizing the total costs (obtained by summing the test and misclassification costs) to the test set size. Test costs are specified as attribute – cost value pairs. The classification costs are defined by the cost matrix $M = (C_{ij})_{nxn}$, where C_{ij} – the cost of misclassifying an instance of class i as being of class j. If the same attribute is tested twice along the path (numeric attribute), the second time its cost is 0.

The particularity presented by ICET, of allowing the test costs (encoded inside a genetic individual) to vary freely in the search domain and then applying the fitness evaluation to guide the individuals towards an optimal solution, increases the variability in the heuristic component. Moreover, *w* and *CF* – two key features in the form of the decision tree – are also varied by the evolutionary component, providing even more possibility of variation in the decision trees search space. Theoretically, this variability is desirable, especially for greedy algorithms such as decision tree learners – that yield unique structures for a fixed training set. Figure 6.1 presents a sketch of the algorithm flow. The genetic algorithm (GA) begins with a set of randomly generated individuals, whose fitness is evaluated by first running EG2 on a training set, using the biases generated by the GA, and then computing the average cost of the generated tree. Then, for a specific number of iterations, it evolves new individuals and computes their fitness. After the last iteration, the fittest individual is returned – its biases are used to train the output classifier.

6.3.2 Enhancements on ICET

A significant problem related to the *ICET* algorithm is rooted in the fact that costs are learned indirectly, through the fitness function. Rare examples are relatively more difficult to be learned by the algorithm. This fact was also observed in [Tur95], where, when analyzing complex cost matrices for a two-class problem, it is noted that: "*it is easier to avoid false positive diagnosis [...] than it is to avoid false negative diagnosis [...]. This is unfortunate, since false negative diagnosis usually carry a heavier penalty, in real life".* This phenomenon is attributed to the distribution of positive and negative examples in the training set. In this context, our aim is to modify the fitness measure as to eliminate such undesirable asymmetries. Last, but not least, previous *ICET* papers focus almost entirely on test costs and lack a comprehensive analysis of the misclassification costs component. We attempt to fill this gap, by providing a comparative analysis with prominent classic approaches for cost-sensitive learning, such as AdaBoost.M1 and MetaCost.

We have considered a series of enhancements to the basic ICET algorithm as well. For each individual, the *n + 2* chromosomes are defined (*n* being the number of attributes in the dataset, while the other two correspond to parameters *w* and *CF*); each chromosome is represented as a 14 bits binary string, encoded in Gray. Gray coding avoids a situation which appears in regular binary coding, the "Hamming Cliffs", in which individuals with similar phenotypes possess significantly different genotypes. The population size has been increased to 50 individuals, since it has a significant impact on the quality of the solution [Kol06]. The *rank-based fitness assignment* technique is used for parent selection, i.e. the fitness assigned to each individual for parent selection depends only on the individual's position in the ranked population, and not on the absolute fitness value. This mechanism ensures uniform scaling across the population and reduces the selective pressure, by limiting the reproduction range, so that no individuals generate an excessive number of offspring [Bac91]. As a result,

stagnation and premature convergence are avoided. As recombination techniques, *single point random mutation* with mutation rate 0.2, and *multipoint crossover*, with 4 randomly select crossover points are employed, to increase search variability rather than favoring the convergence to highly fit individuals early in the search, thus making the search more robust [Spe91].

The algorithm is run for 1000 fitness evaluation steps or until convergence. Due to the fact that a new generation is evolved using single population, which implements *elitism* implicitly, the final result yielded by the procedure is the best individual over the entire run. This makes the decision on when to stop the evolution less critical. More than that, experiments show that usually the best individual does not change significantly after 800 steps: in more than 90% of the cases the algorithm converges before the 800^{th} iteration, while in the rest of the cases the variations after this point are small (less than 3.5%).

The specific choices for the parent selection and recombination methods are based on the intention to increase the search variability by encouraging the exploration of the search space rather than favoring early convergence to highly fit individuals. Simultaneously, the population size and the number of cycles have been increased, and previous best solutions are maintained in the current population, via *elitism*, to avoid premature convergence. Fitness ranking provides an effective mechanism for controlling the selective pressure.

The heuristic component has been developed by altering the J4.8 decision tree classifier to employ ICF as attribute selection function, same as in EG2.

6.3.3 Experimental Evaluation

The experiments evaluate whether the enhancements considered in the genetic component of the ICET algorithm make it perform better in real cases than other similar algorithms. Moreover, we intend to prove that this implementation is better than the original version of the algorithm. In order to do so, we planned three series of tests: the first one tries to solve a problem related to the asymmetry in tree costs for rare examples; a second set of tests provides a more comprehensive analysis of the misclassification cost component; a third set of tests focus on studying the behavior of the ProICET algorithm in real world problems (medical diagnosis problems), also by comparing it with other prominent algorithms.

The datasets used in the evaluations were obtained from the UCI machine learning data repository website (UCI). All are from the medical field and contain test cost information (available on the website as well). Most of the datasets were also used in the original work on ICET, and are presented in appendix A, towards the end of the present dissertation. Other algorithms included in the evaluations are: MetaCost (MC) and EG2 – as state of the art cost-sensitive algorithms, AdaBoost.M1 (AB) – since it is one of the most prominent classifiers focused on error minimization, with proved performance in real-world scenarios [Bre11], and J4.8 as baseline.

The same GA settings were used throughout the evaluations (except for the ones that were dataset-dependent, i.e. the number of chromosomes in an individual). Therefore, for each individual, the n + 2 chromosomes were defined (n being the number of attributes in the dataset, while the other two correspond to parameters *w* and *CF*); each chromosome is represented as a 14 bits binary string, encoded in Gray. The rest of the setting values are listed in table 6.1.

Since the algorithm involves a large heuristic component, the evaluation procedure assumes averaging the costs over 10 runs. Each run uses a pair of randomly generated training-testing sets, in the proportion 70% - 30%; these ten training-testing sets were employed in the evaluations of all algorithms on a particular dataset. The same proportion of 70/30 is used when separating the training set into a component used for training and one for evaluating each individual (in the fitness function).

Table 6.1 – Genetic component settings

Setting	Value
Population type	Single
Initial population generation	random
Population size	50
Crossover cycles	1000
Parent Selection	Roulette wheel
Recombination Operators	Crossover: multiple random crossover, 4 points Mutation: single point random mutation, 0.2 rate
Fitness function	Average cost
Other	Fitness ranking Elitism

Symmetry through Stratification

A significant problem related to the *ICET* algorithm is rooted in the fact that costs are learned indirectly, through the fitness function. Rare examples are relatively more difficult to be learned by the algorithm. Moreover, in [Wei03] it is shown that there exists a strong connection between classifier performance (in terms of accuracy and AUC) and the class distribution employed during training. In this context, the aim is to assess the impact of an imbalanced class distribution on the misclassification cost and to modify the distribution of negative examples in the training set as to eliminate undesirable asymmetries.

First, an evaluation is performed to check whether by providing a balanced training set, ProICET yields better results on some test set than if it were built on an unbalanced set. If the assumption is true, the problem could be eliminated by altering the distribution of the training set, either by over-sampling, or by under-sampling. This hypothesis was tested by performing an evaluation of the ProICET results on two datasets: the Wisconsin breast cancer dataset and Pima Indian diabetes dataset.

Test costs are set to one (1.0) during individual evaluation in the training stage, in order to avoid over-fitting or degenerate solutions. Since the misclassification costs are the one studied by the procedure, test costs are ignored during the evaluation of the final results.

For the stratified training set, the distribution is altered using random uniform over-sampling with replacement on the negative class. Over-sampling is preferred, despite the increase in computation time, because no information loss occurs while the under-represented (minority) class gains in "visibility". Under-sampling is the practical solution for very large databases. In this situation, over-sampling is no longer feasible, as the time required for the learning phase on the extended training set becomes prohibitive, and, thus, under-sampling should be selected.

The misclassification cost matrix used for this analysis has the form:

$$C = 100 * \begin{pmatrix} 0 & p \\ 1-p & 0 \end{pmatrix}, \tag{6.5}$$

where p is varied with a 0.05 increment. Small values for p mean higher costs for misclassifying positive class instances as being negative (this is actually the cost we want to minimize); in the same way, as p approaches 1, the cost of misclassifying negative instances grows. Although there could be an interest in minimizing this cost also, this is not as important as the first situation, because in most real life problems this case is not encountered. As an example of why the second situation is not encountered in real life, suppose the problem of diagnosing a patient of having some heart disease problem. Obviously, it is far riskier to tell a patient having the disease that he is healthy, than the other way around.

The results for the Wisconsin dataset are presented in figure 6.2. Generally, we observe a decrease in misclassification costs for the stratified case throughout the parameter space (with a few small exceptions). This reduction is visible especially in the left side, where we notice a significant reduction in the total cost for expensive rare examples, which was the actual goal of the procedure. Starting from the assumption that the stratification technique may be applicable to other cost-sensitive classifiers, we have repeated the procedure on the WEKA implementation of MetaCost, (MC) using J4.8 as base classifier. J4.8 was also considered in the analysis, as baseline estimate. The results for this set of tests are presented in figure 6.3. We observe that MC yields significantly higher costs, as the cost matrix drifts from the balanced case.

Another important observation is related to the fact that the cost characteristic in the case of J4.8 is almost horizontal for the normal case. This could give an explanation on the general ProICET behavior, of being insensitive to the particular form of the cost matrix (with a few exceptions for the cases with very unbalanced costs). This behavior changes in the stratified case, where the characteristic for J4.8 presents a positive slope; this means that while obvious improvements are achieved for the rare cases (left part of the diagram in fig. 6.3), the cases that are better represented in the original dataset have a higher cost when stratification is applied (right part of the diagram). This observation could also be extended to the way stratification affects ICET behavior, by making it perform a little worse when the "common" cases have a higher misclassification cost. Once again, this is not a tragedy, since those situations do not appear in real life.

A second batch of tests, whose purpose was to study the stratification effect on a dataset having numeric attributes, was performed on the Pima Indian diabetes dataset. The first one to be tested was again ProICET. The results obtained can be observed in figure 6.4. They indicate a poorer performance for the model obtained on the stratified dataset than for the one trained on the original distribution of instances. One reason for this could be found in the nature of the Pima dataset; because all its attributes are numeric, the tree-based classifier must first pre-discretize the attributes. Since stratification applies oversampling and changes the original distribution in the dataset, it may affect the discretization process also, which will choose other values as breakpoints than those chosen for the normal distribution. Therefore, since both models are evaluated on a test set that preserves the original class distribution, the one that was trained on the normal distribution yields better results.

The results yielded by the two other algorithms on the Pima dataset are similar to those obtained for the Wisconsin dataset; they are illustrated in figure 6.5. In this situation, stratification yields only slightly better results for the rare cases than normal training. Moreover, a few observations have to be made. The first one is related to the fact that for balanced costs, MC performs better in the normal case than in the stratified case. The second one refers to the slope of J4.8's characteristic in the stratified case, which is faster ascending than in the Wisconsin case, while the slope for the normal case remains horizontal. An explanation of why these two things happen could be found again in the fact that the Pima dataset has only numeric attributes (except for the class, which is binary), while the Wisconsin dataset has nominal attributes.

A quick note should be made on the fact that the misclassification cost yielded by J4.8 for the Pima dataset is approximately four times higher than the one obtained on the Wisconsin dataset. A reason for this could be found in the nature of the two datasets, since it is a known fact that classification trees perform slightly worse when numeric attributes are involved.

The behavior of the algorithms is mostly uniform on the two datasets (with the exception ProICET presents for the Pima dataset), in that they present an improvement in the misclassification cost for the rare cases when stratification is applied.

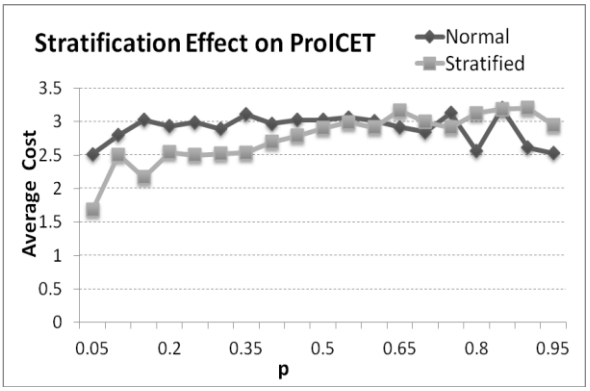


Figure 6.2 – ProICET average misclassification costs for the Wisconsin dataset

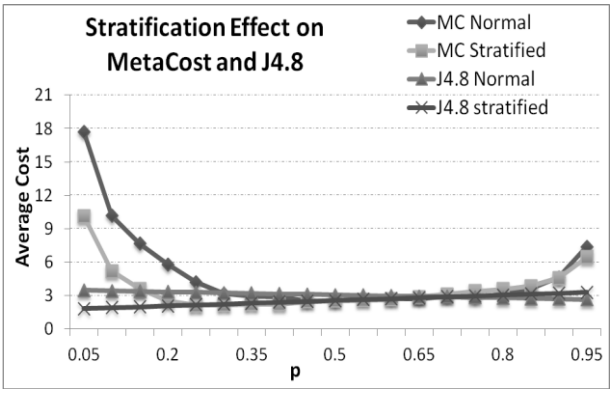


Figure 6.3 – MetaCost and J4.8 average misclassification costs for the Wisconsin Dataset

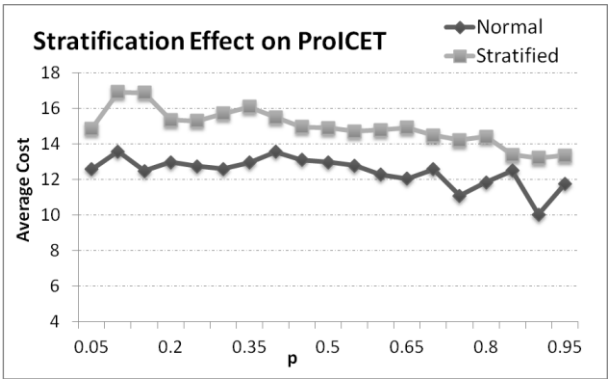


Figure 6.4 – ProICET average misclassification costs for the Pima dataset

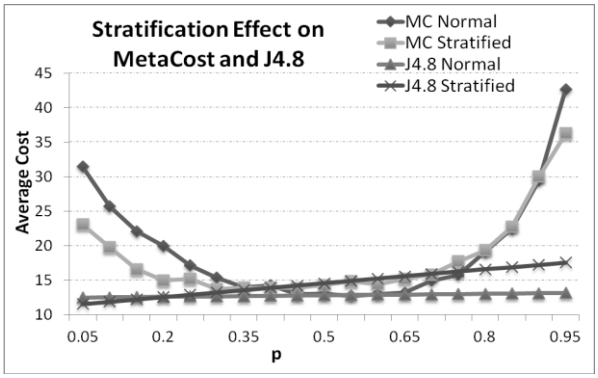


Figure 6.5 – MetaCost and J4.8 misclassification costs for the Pima Dataset

This is a very important result for two reasons: first because of the fact that, although a simple technique, stratification improves ProICET's performance on the cases that are poorer represented in the initial dataset. Secondly, by being algorithm-independent, this technique can easily improve the performance of almost any classifier (exception from this rule are the algorithms that take into account the initial distribution in the dataset, such as AdaBoost.M1 (AB)). Another remark is related to the unexpected results yielded by ProICET on the Pima dataset. Due to this fact, further testing is required before making any general formulations in this matter.

Comparing Misclassification Costs

The procedure employed when comparing misclassification costs is similar to that described in the previous section. Again, the Wisconsin and the Pima datasets were used, and misclassification costs were averaged on 10 randomly generated training/test sets. For all the tests described in this section, the test costs are not considered in the evaluation, in order to isolate the misclassification component and eliminate any bias. However, a test cost of one (1.0) is considered during the training stage of the ProICET algorithm, in order to avoid over-fitted or degenerate solutions.

As illustrated by figure 6.6, for the Wisconsin dataset MC yields the poorest results. ProICET performs slightly better than J4.8, while the smallest costs are obtained for AB, using J4.8 as base classifier. The improved performance is related to the different approaches taken when searching for the solution. If ProICET uses heuristic search, AB implements a procedure that is guaranteed to converge to minimum training error, while the ensemble voting reduces the risk of over-fitting. However, the approach cannot take into account test costs, which should make AB perform worse on problems involving both types of costs.

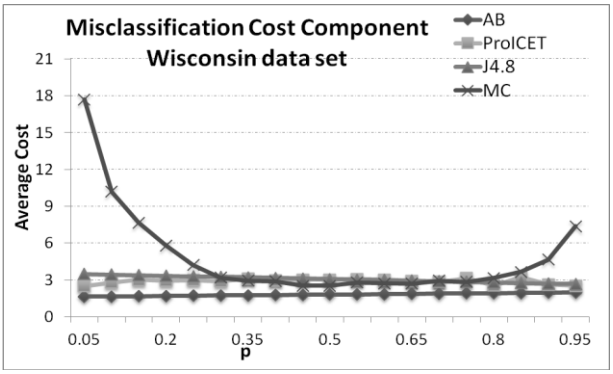


Figure 6.6 – Misclassification costs for the Wisconsin dataset

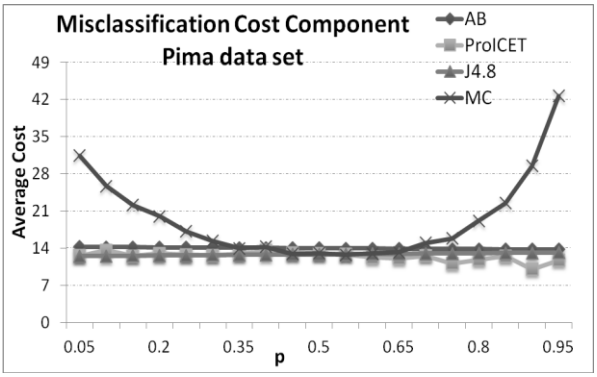


Figure 6.7 – Misclassification costs for the Pima dataset

For the Pima dataset, the misclassification costs obtained by each of the four analyzed algorithms are illustrated in figure 6.7. Again, MC has a very poor performance for the unbalanced cases; ProICET and J4.8 yield similar costs (with ProICET performing slightly better); we can say that these three algorithms have almost the same behavior on the Pima dataset as on the Wisconsin dataset. This is not the case with AB, whose costs are higher than those obtained by ProICET or J4.8.

Total Cost Analysis on Benchmark Medical Problems

When estimating the performance in real life problems of the various algorithms presented, we have considered four benchmark datasets from the UCI repository, representing medical problems: Bupa liver disorders, heart disease Cleveland, Pima Indian diabetes and Thyroid. For the first dataset, we have used the same modified set as in [Tur95]. Also, the test costs estimates are taken from the previously mentioned study. The misclassification costs values were more difficult to estimate, due to the fact that they measure the risks of misdiagnosis, which do not have a clear monetary equivalent. These values are set empirically, assigning higher penalty for undiagnosed disease and keeping the order of magnitude as to balance the two cost components (the actual values are presented in Appendix A, tables A.6.2-A.6.5, along with the attribute test costs).

Since, as far as we know, there exists no other algorithm that would be sensitive to both misclassification and test costs, ProICET had to be evaluated against algorithms which either take at least one of the cost components into consideration, or yield very good error estimates. Bearing this in mind, four algorithms were selected: EG2 as a representative of the algorithms that consider test costs, MetaCost (MC) as a scheme that considers misclassification costs, AdaBoost.M1 (AB) as a very successful ensemble learning method, and J4.8 as a baseline performance measure. We expect ProICET to outperform the other four algorithms since it induces higher variability in the trees search space, and guides that variability into the right direction by using an intelligent evaluation function.

As anticipated, ProICET significantly outperforms all other algorithms, being the only one built for optimizing total costs (*figure 6.8*). Surprisingly, our implementation performs quite well on the heart disease dataset, where the original algorithm obtained poorer results. This improvement is probably owed to the alterations made to the genetic algorithm, which increase population variability and extend the heuristic search. The cost reduction is relatively small in the Thyroid dataset, compared to the others, but is quite large for the two cases, supporting the conclusion that ProICET is the best algorithm for problems involving complex costs. Another remark is related to the fact that, generally, EG2 closely "follows" ProICET's lead, in that its costs are just a little higher than those yielded by ProICET (with one exception for the Bupa dataset). One reason for this would be the relation between test costs and misclassification costs; the misclassification costs were chosen such as to have the

same magnitude order with the test costs that came with the datasets and follow the idea of expensive test costs relative to misclassification costs.

On the Bupa dataset, AB performs slightly better than ProICET, while the other algorithms have significantly poorer performances. This could be a consequence of the relation between test and misclassification costs; for the Bupa dataset the test cost of misclassifying a negative case as positive is lower than any attribute test cost, and the other misclassification cost has the cost of approximately 2 attributes (see Appendix A for details); this particularity in the cost setting procedure makes the algorithms that are best at error reduction perform better than the others (this is the case with AB). Still, a note should be made on the fact that on this dataset, ProICET achieves the second best performance. Moreover, the next tests will show that, for the other datasets, ProICET outperforms all the considered algorithms.

On the Thyroid dataset, ProICET achieves the best cost. Still, this error reduction is not as clear as in the other datasets. The fact that the costs are so uniform for this dataset may be rooted in the size of the dataset – 7200 instances. Although it is a rather difficult "to learn" dataset, because it has many attributes, of which several are numeric, the dimension of the set seems to compensates for that complexity.

As illustrated in figure 6.8 (c), on the Cleveland dataset ProICET outperforms by far MC, and the error reduction algorithms (AB and J4.8), while EG2 yields a very good cost also. This is a rather surprising result, since in the initial paper ICET obtained very poor results when tested on this dataset. The reasons could be rooted in the fact that the number of iterations for the individual training in the genetic component was significantly smaller, and standard values for GA parameters were used, which proved to be not so helpful; also, a 50/50 split was used for the sub-training and sub-testing sets during the training stage, therefore drastically affecting EG2's capacity of building successful trees.

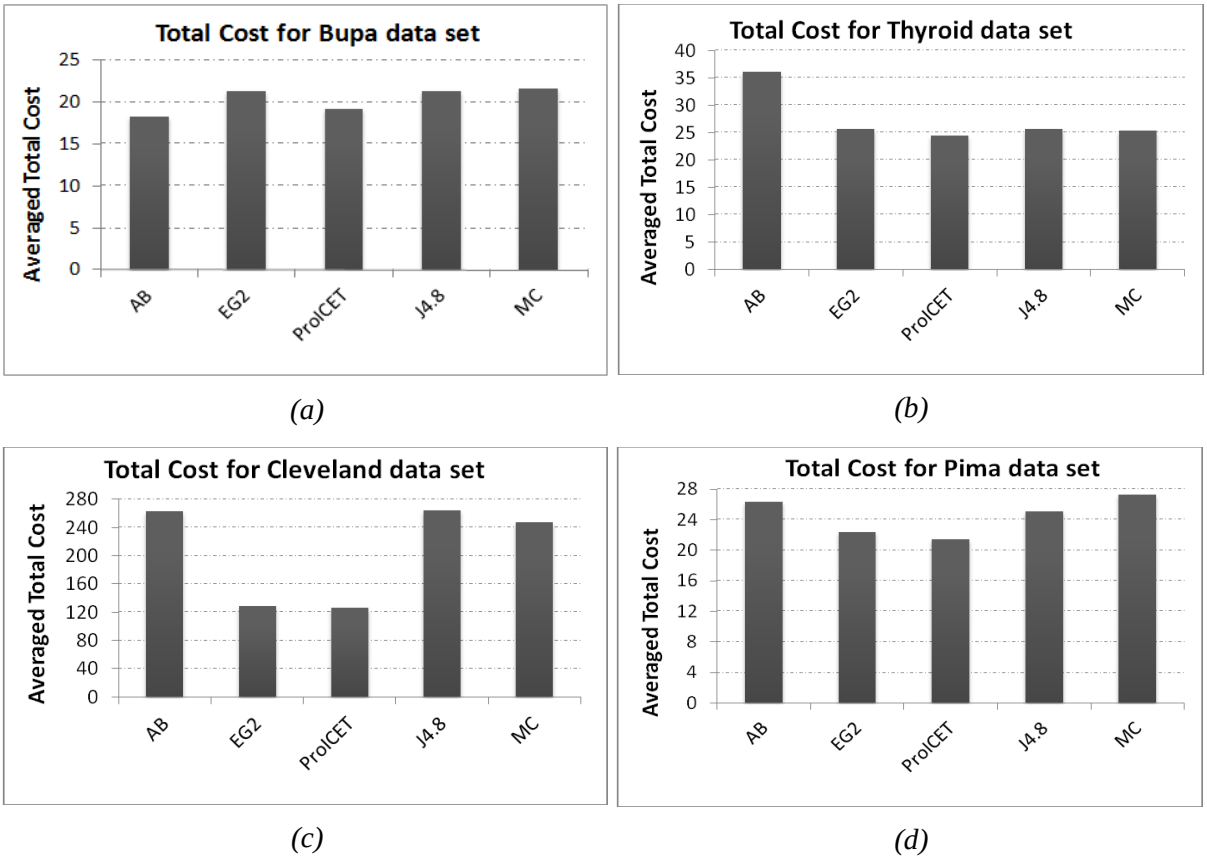


Figure 6.8 – Total costs obtained by the classifiers on different benchmark medical problems

The results on the Pima dataset are illustrated in figure 6.8 (d). Although the cost distribution for the Pima dataset is rather uniform, ProICET yields visibly lower costs. Again we notice that EG2 is the second in line, while the others have higher costs.

6.3.4 Case Study: ProICET on a Prostate Cancer Problem

Medical data mining is considered to be one of the most challenging areas of application in knowledge discovery. Main difficulties are related to the complex nature of data (heterogeneous, hierarchical, time series), or to its quality (possibly many missing values) and quantity. Domain knowledge or ethical and social issues are also of great importance. But maybe the most important particularity of medical data mining problems is the concept of cost.

When mining a medical problem, the concept of cost interferes in several key points. First of all, a doctor must always consider the potential consequences of a misdiagnosis. In this field, misclassification costs may not have a direct monetary quantification, but they represent a more general measure of the impact each particular misclassification may have on human life. These costs are non-uniform (diagnosing a sick patient as healthy carries a higher cost than diagnosing a healthy patient as sick). Another particularity of the medical diagnosis problem is that medical tests are usually costly. Moreover, collecting test results may be time-consuming. Arguably, time may not be a real cost, but it does have some implication for the decision whether it is practical to take a certain test or not. In the real case, performing all possible tests in advance is unfeasible and only a relevant subset should be selected. The decision on performing or not a certain test should be based on the relation between its cost and potential benefits. When the cost of a test exceeds the penalty for a misclassification, further testing is no longer economically justified.

Since ProICET considers both test and misclassification costs, and employs a promising learning strategy, it has the potential for providing a successful solution in data mining problems involving medical diagnosis. This section presents a case study of applying the ProICET algorithm on a prostate cancer diagnosis problem. The current research has been performed within the CEEX research grant no. 18/2005 – IntelPRO (Intelligent system for assisting the therapeutic decision at patients with Prostate cancer)

Prostate cancer occurs when cells of the prostate (gland in the male reproductive system) mutate and begin to multiply out of control, spreading to other parts of the body (bones and lymph nodes mainly). Symptoms include pain, difficulty in urinating, erectile dysfunction, and many others. Physical examination and PSA (Prostate Specific Antigen) blood tests are crucial for early diagnosis of the disease. Confirmation is received upon performing a biopsy of the prostate tissue. Further investigations, such as X-rays and bone scans, may be performed to determine the degree of spread. There are several possible treatments for prostate cancer, such as: surgery, radiation therapy, chemotherapy, hormone therapy, or a combination of these – depending on the extent of spread of the disease, age and general state of health, and so on. The medical tests performed during the diagnosis of the disease – especially the biopsy – are costly. Also, it is evident that missing to identify a positive diagnosis is far more harmful than the reverse error.

The main goal of applying to the prostate cancer diagnosis problem has been to verify that it maintains its behavior in this real-world medical problem, yielding low costs while maintaining a high precision rate. A second direction of investigation involved ranking the predictor attributes, such as to try and match results obtained by the data mining system with the medical staff assumptions. The dataset has been provided by the Medicine and Pharmacy University of Cluj-Napoca. During the discussions with the medical team, a set of major/immediate interest parameters were defined (presented in Appendix A, table A6.6). The same evaluation procedure was employed as in the previous section (repeated percentage split). Two different values for the test costs were used – 0 and 0.1 – and four different cost

matrices (built such as to emphasize the unbalance in different errors' severity) – also presented in Appendix A, table A6.6. This resulted in eight different evaluation scenarios. The results are presented in table 6.2.

It can be observed that, when both types of costs are considered, *ProICET* yields the lowest total costs, which proves once again it is the best approach for cost reduction in medical problems. The fact that the accuracy rates (~84%) do not reach very high values could be rooted in the characteristics of the dataset: the number of instances was reduced during the pre-processing stage, because of the high number of missing values.

Another important result is related to the ranking of the attributes in the order of their prediction power. Since during the training process equal test costs were assigned to each attribute, the cost component did not influence in any way the choice of one attribute over another (it only affects the total cost of the trees in the sense that bigger trees yield higher total costs). By analyzing the output the following attributes resulted as possessing the highest prediction power (the same list was obtained by the other algorithms as well):

- Prostate Volume
- Operation Technique
- Bleeding
- Gleason Score
- IIEF
- Pre-Op PSA

The fact that the prostate volume appears the first in most tests is new, and, according to the medical team's opinion, it is the confirmation of a fact they have been suspecting for some time now.

6.4 Conclusions on Cost-Sensitive Classification

Several efforts have been made in the machine learning community to develop learning schemes that are sensitive to a particular type of cost. Considerably less research has been conducted to develop an algorithm that tackles both costs, the most significant such method being ICET. It has been shown to yield better results when compared to algorithms that are sensitive only to test costs. The initial work on ICET lacked a comprehensive study on the misclassification cost component though, as well as an evaluation of ICET in real problems.

Table 6.2 – Total costs and accuracy rates of the various algorithms on the Prostate Cancer dataset (TC – value of Test Costs; CM – Cost Matrix)

Cost Settings	Average Accuracy Rate (%)					Average Total Cost				
	Pro ICET	***AB***	***EG2***	***J4.8***	***MC***	***Pro ICET***	***AB***	***EG2***	***J4.8***	***MC***
TC:0,CM:1	84.18	79.18	84.07	84.07	84.18	0.28	0.284	0.269	0.269	0.293
TC:0.1,CM:1	83.77					**0.414**	0.734	0.430	0.430	0.448
TC:0,CM:2	83.87				83.26	0.561	0.52	0.52	0.52	0.65
TC:0.1,CM:2	84.07					**0.678**	0.97	0.682	0.682	0.812
TC:0,CM:3	84.28				84.38	0.146	0.166	0.142	0.142	0.145
TC:0.1,CM:3	84.07					**0.252**	0.616	0.305	0.305	0.310
TC:0,CM:4	84.07				83.36	0.213	0.44	0.44	0.44	0.502
TC:0.1,CM:4	83.77					**0.575**	0.89	0.603	0.603	0.647

There are two main particularities of ICET that make it an appropriate for solving cost-sensitive problems:

- It considers both test costs and misclassification costs, while other cost-sensitive algorithms considered just one of these components
- The strategy it uses, of combining evolutionary methods with classical top-down inductive search (a greedy search), is very fresh and promising, because it introduces an element of variability in the search space where the greedy component performs

The main contributions presented in this chapter focus on a series of improvements that could be added to the initial algorithm, as well as filling in the gap in the algorithm evaluation part. Consequently, a new version of the algorithm has been developed, ProICET, which employs the basic technique proposed by ICET, while improving the strategies used in specific parts. A series of evaluations which focused on assessing the performance of ProICET under a series of different aspects have been performed.

The first set of tests was focused on finding the validity of a theory formulated in the initial work on ICET, concerning the asymmetry in the misclassification costs component – that for the "rare" cases, both ICET and the other algorithms considered yielded higher costs. Stratification was proposed there as a theoretical solution, but no tests were performed to try and check the hypothesis' validity. Therefore, the current dissertation attempts to prove empirically that the theory holds, and, moreover, that it is extendable to other schemes. The second batch of tests was aimed at analyzing the misclassification cost component. A third series of tests were developed to evaluate the behavior of ProICET in real medical diagnosis problems, which contained both types of costs.

The results allow for the formulation of the following conclusions:

- ProICET outperforms its direct competition (algorithms that are also sensitive to some kind of cost). This improvement can be seen in the evaluation of the misclassification cost component, where ProICET performs better than the other algorithms, with one single exception for the Wisconsin dataset, where it yields higher costs than AdaBoost.M1. In the tests that have been performed on benchmark medical, ProICET yields constantly better costs than algorithms that consider only test costs, or algorithms that are aware to misclassification costs. Most importantly, when evaluated on a real prostate cancer problem, ProICET always yielded lowest total cost of all the algorithms considered, when both test and misclassification costs were included, keeping a high level of accuracy.
- This particular implementation has proved to be even more successful than the original version of the algorithm, because it yields better results on the Cleveland heart disease dataset, where the initial implementation obtained rather modest results. This improvement is due to the modifications made in the evolutionary component: the introduction of elitism, increased search variability factor, and extended number of iterations. Also, increasing the size of the training set proved to be successful.
- Stratification generally produces improvements in the misclassification cost of rare cases, for all classifiers; an exception to this rule is observed for ProICET and J4.8 in the case of datasets with numeric attributes – the pre-discretization process affects stratification. A note should be made on the fact that for the stratification problem, ProICET is almost symmetric, in that it is insensitive to the particular form of the cost matrix. This could be a consequence of the fact that the tree learner is a modified version of J4.8 (which displays a horizontal characteristic).

The original work presented in this chapter comes as a result of the involvement in the CEEX research grant no. 18/2005 – IntelPRO (Intelligent system for assisting the therapeutically decision at patients with Prostate cancer). The ProICET algorithm has been

applied to a real problem related to prostate cancer diagnosis and prognosis: predicting the value of post-operative PSA for patients who have undergone prostate surgery, from data recorded pre- and immediately after the operation. The results confirmed yet again that the method provides the most suitable approach in such problems, yielding lowest total costs and high accuracy rates under different evaluation scenarios. Also, the ranking of predictive attributes produced a new piece of information, which confirmed (previously un-validated) medical team assumptions.

The results presented in this chapter have been disseminated and accepted by the research community through the publication of 1 journal paper (in *Health Informatics Journal*):

1. **C. Vidrighin Bratu** and R. Potolea, "ProICET: a cost-sensitive system for prostate cancer data", *Health Informatics Journal*, Dec 2008, vol. 14: pp. 297-307, ISSN: 1741-2811 (online); 1460-4582

1 book chapter (in *Advances in Greedy Algorithms*):

1. **C. Vidrighin Bratu** and R. Potolea, "Enhancing Greedy Policy Techniques for Complex Cost-Sensitive Problems", in *Advances in Greedy Algorithms*, IN-TECH, 2008, pp. 151-168, ISBN 978-953-7619-27-5 (print)

and 3 conference papers, presented in the proceedings of established international conferences:

1. **C. Vidrighin**, R. Potolea, I. Giurgiu, M. Cuibus, "ProICET: Case Study on Prostate Cancer Data", *Proceedings of the 12th International Symposium of Health Information Management Research*, 18-20 July 2007, Sheffield, pp. 237-244
2. R. Potolea, **C. Vidrighin**, C. Savin, "ProICET - A Cost-Sensitive System for the Medical Domain", *Proceedings of the 3rd International Conference on Natural Computation ICNC 2007*, Haikou, August 2007, China, Volume 2, Session 3
3. **C. Vidrighin Bratu**, C. Savin, R. Potolea, "A Hybrid Algorithm for Medical Diagnosis". *Proceedings of the IEEE The International Conference on Computer as a Tool*, 9-12 September 2007, Warsaw, pp. 668-673

7 Classification in Practice II: Imbalanced Class Distribution

7.1 Problem Statement

One of the current important challenges in data mining research is classification under an imbalanced data distribution. This issue appears when a classifier has to identify a rare, but important case. Traditionally, domains in which class imbalance is prevalent include fraud or intrusion detection, medical diagnosis, risk management, text classification and information retrieval [Cha04]. More recent reports include unexploded ordnance detection [Ali06], or mine detection [Wil09].

A classification problem is imbalanced if, in the available data, a certain class is represented by a very small number of instances compared to the other classes [Jap02]. In practice, the problem is addressed with 2-class problems; multi-class problems are translated to binary. As the minority instances are of greater interest, they are referred to as positive instances (positive class); the majority class is referred to as the negative class.

7.1.1 Imbalance-Related Factors

The first step in providing viable solutions for imbalanced domains is to understand the problem: what is the real issue with the imbalance? Initially, the difficulty of dealing with imbalance problems was thought of coming from its *imbalance rate* (IR), i.e. the ratio between the number of instances in the majority (m_{Maj}) and minority classes (m_{Min}):

$$IR = \frac{m_{Maj}}{m_{Min}} \tag{7.1}$$

More recent studies suggest that the nature of imbalanced problems is actually manifold. In [Wei04], two issues are considered as being crucial: (1) insufficient data to build a model, in case the minority class has only a few examples (similar to dealing with small samples/small datasets), (2) too many "special cases" in the minority class, so that in the class itself, some kind of sub-clustering occurs, which might lead again to insufficient examples for correctly identifying such a sub-cluster. These two cases translate into two *types of rarity*: *between-class* (1) vs. *within-class* (2). While the between-class imbalance faces the issue of a peculiar class distribution only, for which some intelligent sampling techniques could help [Cha98], within class imbalance is trickier. Besides an increased complexity of the data (which implies that the model should identify a rule for each sub-cluster), the small sample problem could override it, which hinders the identification of each sub-cluster. The between class imbalance is also referred to as *rare class*, while within-class as *rare case*. For the within class imbalance, a special case is represented by the *small disjuncts problem* [Hol89]. It has been observed that, in most cases, imbalanced problems suffer from the small disjuncts problem – the existence of "isolated" subsets of only a few instances in the minority class, surrounded by instances from the other class(es), making them difficult to identify [Wei04]. Ideally, a concept is best identified when it can be defined as a purely conjunctive definition. In real settings, for complex concepts this is not always possible. Therefore, a concept is defined by several disjuncts, each being a conjunction expressing a sub-concept. In many cases, some of those disjuncts have small coverage, and are therefore difficult to identify. Small disjuncts are much more error prone [Hol89] than large disjuncts. *Dataset shift* [Ala08] and *class overlapping* [Den10] have also been recently identified as being important factors related to the imbalance.

An important theoretical result related to the nature of class imbalance is presented in [Jap02], where it is concluded that the imbalance problem is a relative problem, which depends on: (1) the imbalance ratio, i.e. the ratio of the majority to the minority instances, (2) the complexity of the concept represented by the data, (3) the overall size of the training set

and (4) the classifier involved. The experiments there were conducted on artificially generated data, in the attempt to simulate different imbalance ratios, complexities and dataset sizes. The concept complexity has been approximated through the size of the decision tree generated on the data (as $\log_2(no.leaves)$). The results have indicated that C5.0 is the most sensitive learner to the imbalance problem, while the Multilayer Perceptron showed a less categorical sensitivity pattern and the Support Vector Machine seemed to be insensitive to the problem.

In [Pot11c] we have extended the analysis by performing a set of experiments on benchmark datasets, to study the effect of the class imbalance problem on a broader spectrum of algorithms. An initial study focused on the factors described in [Jap02] – dataset size, imbalance ratio, complexity and learning algorithm, in an attempt to address some of the open questions presented in the above mentioned work, related to the applicability of the conclusions drawn on artificial data in real-world settings. The results (which are detailed in section 7.1.3) suggest that a more meaningful analysis can be performed by considering IR and a new meta-feature, which combines data size and complexity information. The *instances-per-attribute ratio* (IAR), i.e. the ratio between the total number of instances (*m*) and the number of attributes recorded per instance (*n*) is more significant than the separate size and complexity measures, allowing for a faster and easier initial assessment of a particular dataset:

$$IAR = \frac{m}{n} \tag{7.2}$$

7.1.2 Estimating Performance

Establishing how to assess performance is an essential task in imbalanced problems. The selection of an inappropriate evaluation measure may lead to unexpected predictions, which are not in agreement with the problem goals.

The most widely employed metric in the early (theoretical) stage of data mining research was the *accuracy* (Acc) of the classifier. Even today it is widely employed when assessing the performance of new learning schemes, being an appropriate metric even for real-world, balanced problems. When dealing with an imbalanced problem, however, it provides an insufficient measure of the performance [Wei04, Cha06], because the minority class contributes very little to its value. In highly imbalanced problems, a good recognition of the majority class will translate into a high accuracy, regardless of how well the model identifies minority cases. Thus, for a dataset with 99% examples for one class and 1% for the other, a model which classifies everything as belonging to the majority class will yield 99% accuracy, while failing to identify any minority example.

Therefore, the evaluation of imbalanced problems requires other metrics which provide a more directed focus. Such a metric, which focuses on the recognition of the minority class, is the TP_{rate} (sensitivity/recall). Generally, the TN_{rate} (specificity) is not so important in imbalanced problems [Grz05]. On the other hand, in some situations it is important to "improve recall without hurting precision" [Cha06]. Therefore, besides sensitivity, precision may also have an important role when dealing with such problems. Controlling the relative importance between precision and recall is another strategy which could provide a correct assessment in imbalanced scenarios, by employing a precision/recall curve, or the F_i-value – which can be tuned to put more emphasis on either the recall or precision: $i > 1$ for when recall is more important. In certain situations, besides TP_{rate}, keeping a high TN_{rate} may be important. For such situations, equidistant metrics, such as the geometric mean or the balanced accuracy (defined in Chapter 2) provide appropriate performance assessment.

Thus, according to the specifics of the problem at hand, one should carefully assess which metrics to consider. In many information extraction applications, for example, the f-measure is considered to offer the best trade-off between precision and recall, since it is desired to detect as many positive items as possible, without introducing false positives. On the other hand, in medical diagnosis, it is essential to identify all positive cases, if possible, even at the risk of introducing false alarms (which may be eliminated through additional medical investigations). The same occurs in fraud detection, where the cost of missing to identify a fraud is so high that a certain level of false positives is acceptable. The opposite situation can also appear. In credit risk assessment, for example, introducing false positives is unacceptable. However, a mild decrease of the number of identified positive cases is usually acceptable, since it is preferable to lose a potential client in the attempt to avoid a default. Thus, there are situations in which maximizing the TP_{rate} is of utmost importance, situations in which precision must be kept at high levels, even at the cost of mildly decreasing the TP_{rate}, or situations in which both are equally significant.

In view of what has been presented, we argue that metric selection in imbalanced problems is essential for both model quality assessment and guiding the learning process. The metric should also reflect the goal of the specific classification process, not just focus on the data imbalance. Thus, if we are additionally dealing with imbalance at the level of the error costs, then associating a cost parameter to account for such disproportions is appropriate. If, on the other hand, the focus is on identifying both classes correctly, then an equidistant metric provides a fair estimation.

7.1.3 The Effect of the Class Imbalance on the Performance of Classifiers

In order to study the nature of the imbalance problem, we have considered 34 datasets from the UCI machine learning data repository (Appendix A, table A.7.1). A number of problems were modified to obtain binary classification problems from multi-class data. Learning algorithms belonging to 6 different classes were considered: instance based learning – kNN (k Nearest Neighbor), Decision Trees – C4.5, Support Vector Machines – SVM, Artificial Neural Networks – MLP (Multilayer Perceptron), Bayesian learning – NB (Naïve Bayes) and ensemble learning – AB (AdaBoost.M1). We have employed the implementation in the WEKA framework for the six methods selected, and their default parameter values. The evaluations were performed using 10-fold cross validation, and reporting the average values obtained. The following metrics were recorded: the accuracy (Acc), TP_{rate}, and TN_{rate}. Also, the geometric mean (GM), the balanced accuracy (BAcc) and the f-measure (FM) have been computed. The minority class in all problems is the positive class. An initial analysis was carried out on the data grouped by size, IR and complexity (C), into the categories presented in Table 7.1. Not all combinations of the three categories can be found in the datasets we have evaluated: for example, a very large complexity is only represented in the large datasets category.

Table 7.2 presents a summary of the results obtained by the learning algorithms on the different categories of problems. Shaded rows represent data categories sensitive to imbalance, while non-shaded rows represent groups of problems on which classifiers have a robust behavior, under TP_{rate}. We have selected this metric to assess robustness since, as suggested in [Jap02], performance degradation is related to a large drop in the TP_{rate}.

Table 7.1 – Dataset grouping on size, IR, C

Dimension Category	*Very small*	*Small*	*Medium*	*Large*	*Very large*
Size (no. of instances)	<400	400-1500	2000-5000	>5000	-
Rounded IR	-	<9	-	>=9	-
Rounded C	-	<=2	[3,4]	[5,9]	>=10

Table 7.2 – TPrates obtained by classifiers on the different categories of problems

Set Size	IR	Complexity	kNN	C4.5	SVM	MLP	NB	AB
very small	*<9*	*Small*	.53	.5	.5	.61	**65**	.57
		Medium	**.72**	**.71**	.3	.61	.65	.65
		Large	.73	.72	.79	.76	**.8**	**.81**
	>=9	*Medium*	.52	.6	.15	.59	**.83**	.4
small	*<9*	*Medium*	.88	**.89**	**.89**	**.9**	**.89**	.83
		Large	.81	.77	**.85**	.81	.62	.67
	>=9	*Medium*	**.98**	.94	**.98**	**.99**	**.98**	**.99**
		Large	.24	.09	.47	**.65**	.09	.0
medium	*<9*	*Large*	.74	**.97**	.92	**.98**	.69	.85
	>=9	*Medium*	.6	**.91**	.5	.86	.78	.89
		Large	.57	**.88**	.04	.73	.84	.82
large	*<9*	*Large*	**1**	**1**	**1**	**1**	.92	.98
	>=9	*Very Large*	.06	.0	.01	.0	**.39**	.0

Also, for each dataset category the best performance and the worst performance have been marked (bold-face and underline, respectively). The results agree with the conclusions presented in [Jap02] that the value of the IR plays an important role in the performance of the classifiers. However, an increase in the complexity does not necessarily lead to classifier performance degradation, as the results for the [*IR<9, size – very small*] category indicate. Moreover, size and complexity are related, since, the size increases, the data exhibits higher complexity.

As it can be observed from figures 7.1 – 7.4, the behavior of classifiers on large complexity datasets is better than on categories of problems of smaller complexity (in fig. 7.3 almost all classifiers seem to be robust to the imbalance problem). Still, for the other set size categories (small, medium and large), a large imbalance (IR>=9) associated with increased complexity (large, large and very large) always affects the learning process (Table 7.2).

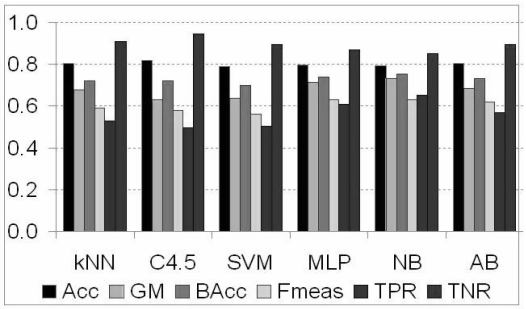


Figure 7.1 - Size very small, IR<9, C small

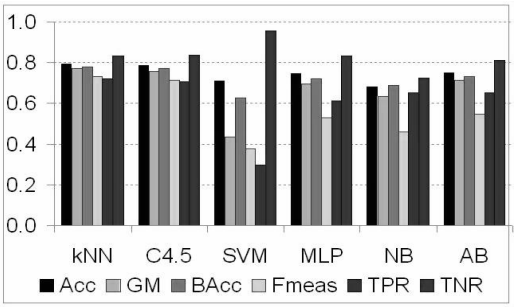


Figure 7.2 - Size very small, IR<9, C medium

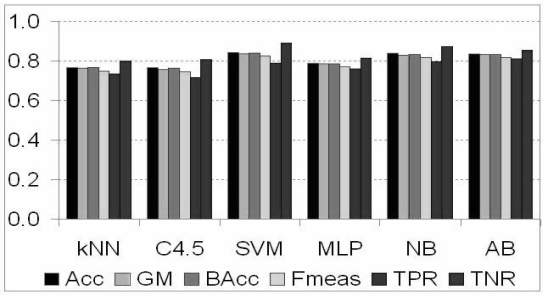


Figure 7.3 - Size very small, IR<9, C large

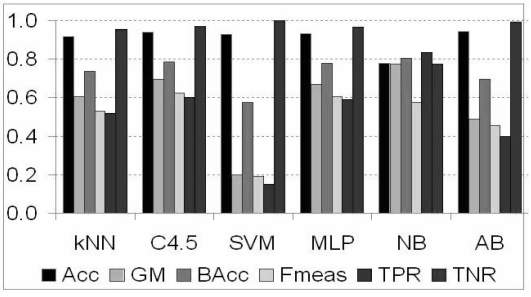


Figure 7.4 - Size very small, IR>=9, C medium

The results suggest that neither dataset size, nor the complexity alone represent strong (monotonic) indicators of the IR's influence in the classification process. We consider that poor concept identification is related to the lack of information caused by insufficient examples to learn from. However, a relation between problem size, complexity and classifier performance is revealed, i.e. the larger the dataset size, the higher the complexity for which the performance degradation becomes clear. This suggested the existence of another meta-feature which better discriminates the classifier robustness when faced with imbalanced problems, the instance per attribute ratio (IAR).

The diagrams in figures 7.5 – 7.7 present the performance of the same classifiers, under different metrics, on the problem categories which affect their learning capacity. The accuracy alone is not a good measure of performance. The analysis should focus on the following criteria: high values for TP_{rate}, GM, BAcc and Fmeasure indicate a good classification, while high TN_{rate} values reveal a classification which is biased towards the majority class. Moreover, the larger the difference between the TN_{rate} and the TP_{rate}, the more biased the classification process is.

The results prove that the learning capabilities of the classifiers considered are affected to some extent by an increased imbalance in conjunction with the other data-related particularities. It can be observed that, like in [Jap02], MLPs are generally more robust than C4.5 to the imbalance problem. Moreover, they are the least affected by the imbalance-related factors, in most cases. As an exception, C4.5 performs noticeably better than MLP (and all the others, actually) on medium sized datasets, with large IR and C (fig. 7.6). The analysis also reveals that the NB classifiers have a good general behavior when dealing with a very large imbalance. In some cases they even yield the best performance (figures 7.1, 7.4, 7.7 – all with IR>=9). However, they are not as robust as MLPs, since, in some cases, they achieve a very poor performance (fig. 7.5). Although not always the best classifier, MLPs yield at least the second best performance in all cases, which makes them the most robust out of all the classifiers evaluated. None of the kNN and AB show remarkable results in any of the cases studied, which makes them suitable only for baseline problem assessment.

The above observations provide an affirmative answer to one of the open questions in [Jap02], whether the conclusions presented there can be applied to real-world domains. However, our results also indicate that SVM are the most sensitive to imbalance. This means that, for the particular case of SVMs, the conclusion drawn from experiments on artificial data cannot be extended to real datasets. A justification for this could be the following: in the case of artificial datasets, even for large IRs, the examples which represent reliable support vectors are present in the data, due to the systematic data generation process, while in the case of real problems, these vital learning elements might be missing. This makes SVMs the weakest classifiers in most real-world imbalanced problems.

We have performed a second analysis for studying the effect of imbalanced problems on the performance of the classifiers, using another dataset grouping: by IR and by the ratio between the number of instances and the number of attributes (IAR).

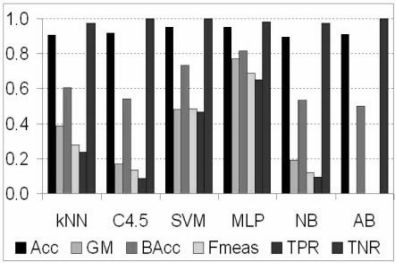


Figure 7.5 - Size small, C large

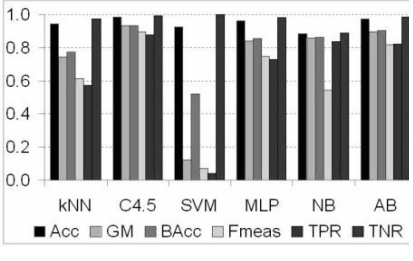


Figure 7.6 - Size med., C large

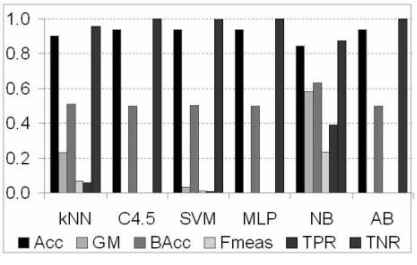


Figure 7.7 - Size large, C v. large

Table 7.3 - Dataset grouping on IR, IAR

Parameter	***Category***	***Value Range***
Rounded IR	*Balanced*	~1
	Small	[2,3]
	Large	>=4
Rounded IAR	*Small*	<=60
	Medium	(60, 100]
	Large	(100, 200]
	Very large	>200

Table 7.4 - TPrates on IR and IAR grouping

IR	**IAR**	**kNN**	**C4.5**	**SVM**	**MLP**	**NB**	**AB**
Balanced	*Small*	.68	.71	.72	.7	.58	**.75**
	Medium	**.94**	**.95**	.8	.86	.78	.85
	Very large	**1**	**1**	**1**	**1**	.92	.98
Small	*Small*	.71	.69	.53	.72	**.78**	.65
	Medium	.81	.77	**.82**	**.83**	.67	.63
Large	*Small*	.5	.55	.27	.62	**.64**	.4
	Medium	.53	.52	**.72**	**.73**	.59	.49
	Large	.58	**.89**	.19	.74	.82	.84

We consider this new meta-feature successfully combines size and complexity information: a small IAR should yield a higher classifier sensibility to the imbalance problem, while a very large IAR should provide more robustness to the imbalance. The categories for this second analysis are summarized in Table 7.3. By re-grouping the evaluations according to this new criterion, we noticed a more clear separation between the different categories and that classifiers better learn with larger IARs. Indeed, as we can observe from Table 7.4, the larger the IAR, the larger the IR for which the TP_{rate} value of the classifiers decreases. Also, for the same IR, as IAR increases, classifiers are more robust to the imbalance. The different levels of shading used for the rows indicate the performance level (more shading, better average performance). Again, we have marked the highest and lowest TP_{rate} values for each problem category (bolded and underlined, respectively).

Figures 7.8 – 7.11 present the performance of the classifiers under this second categorization, for all metrics considered, on the relevant groups (problems which are affected the most by the imbalance related issues). The diagrams indicate again that SVM are unstable classifiers for imbalanced problems (strongly biased towards the majority class). Out of all classifiers, MLP are the most robust, yielding either the best or second best performance. The NB classifier generally achieves the best recognition of the minority class (maximum TP_{rate}). However, it is not the best classifier due to poor recognition of the majority class (lowest TN_{rate} in all cases). This makes the NB classifier the most appropriate for imbalanced problems in which the minority class possesses a significantly larger importance than the majority class. Similar to the previous analysis, kNN and AB have a variable behavior, which hinders the identification of a situation in which they could guarantee quality results. If we have found that a large IAR improves the behavior of classifiers for the same IR, it appears that C4.5 is the most responsive to a large IAR, as it can be observed from fig. 7.11. All the above measurements refer to pruned versions of C4.5.

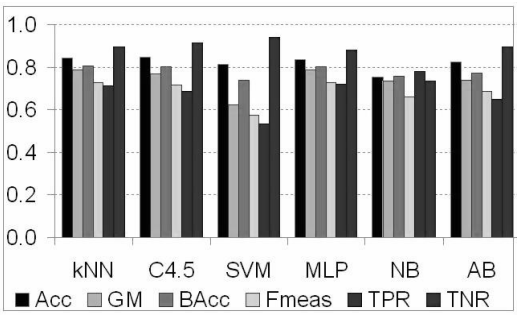


Figure 7.8 - IR small imbalance, IAR small

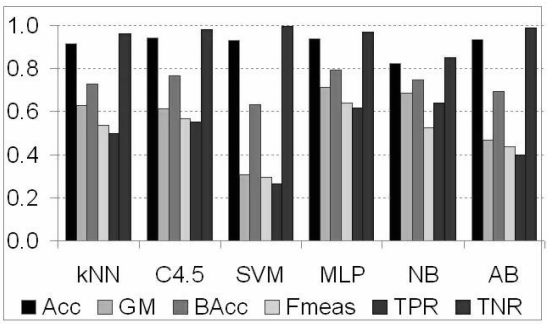


Figure 7.9 - IR large, IAR small

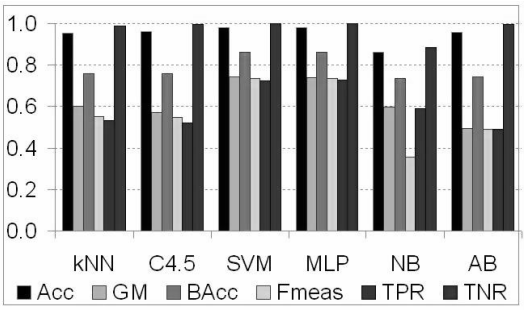


Figure 7.10 - IR large, IAR medium

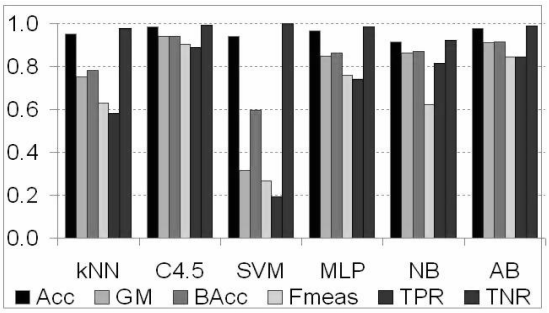


Figure 7.11 - IR large, IAR large

In [Jap02], it is argued that, for large IRs, unpruned C4.5 models are better than the pruned versions. We have performed an evaluation to validate this statement, using the Mushrooms benchmark problem – large size, balanced dataset – by varying the IR up to 100. The evaluation was performed in a 10-fold cross validation loop. The results are presented in the diagrams from fig. 7.12. We have employed the logarithmic scale for the horizontal axis (IR), to better differentiate between the two curves at smaller IRs. By comparing the two diagrams we notice that GM is more fitted for this situation, as it is more realistic in estimating the performance (BAcc being overoptimistic), and it better differentiates between the pruned/unpruned versions. This is due to the fact that a larger difference between two variables is more visible in the product than the sum of their values.

On the same relatively large dataset (Mushrooms), a series of experiments have been conducted to study the effect of varying IR and IAR on the performance of the different classifiers. The IAR has been varied via two mechanisms: *(1)* by varying the size of the training set (via random sampling) and keeping the number of attributes constant and *(2)* by varying the number of attributes and apply obtain the maximum possible size for the given IR, IAR and number of attributes (via random sampling).

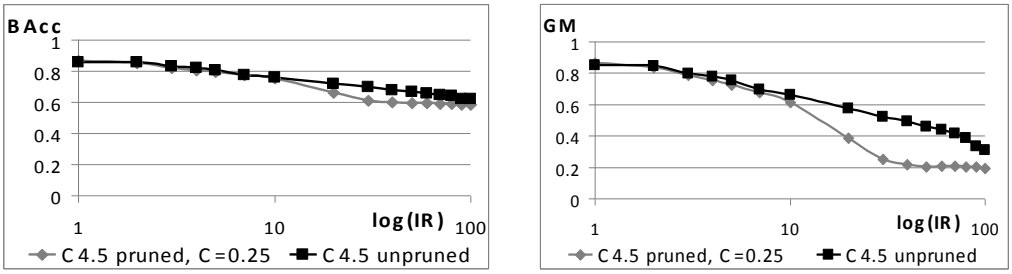
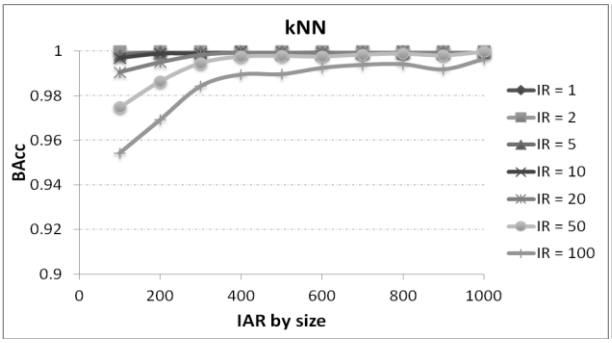
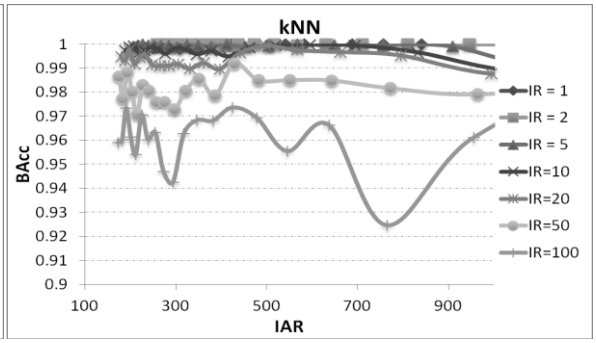


Figure 7.12 - Performance degradation for C4.5 on mushrooms dataset, under the balanced accuracy (BAcc) and the geometric mean (GM)

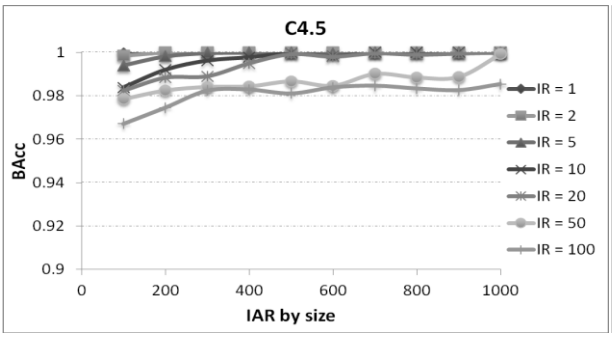
For the second scenario, the attributes have been initially ranked using the gain ratio as measure, and the size of the predictive attribute subsets was varied between 2 and the number of predictive attributes in the dataset (22). The results of these evaluations are presented in diagrams *(a) – (l)* from figure 7.13. The diagrams on the left present the BAcc levels obtained by the different classifiers by varying IR and IAR by size (scenario 1), and the right-side diagrams present the results obtained by varying IR and IAR by the number of attributes (scenario 2).



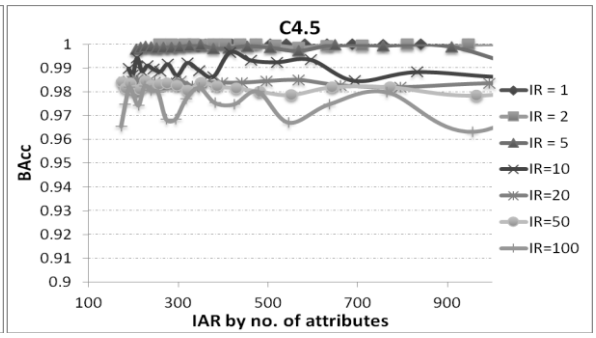
(a)



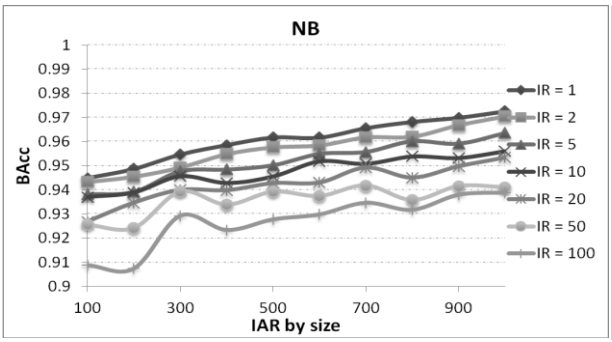
(b)



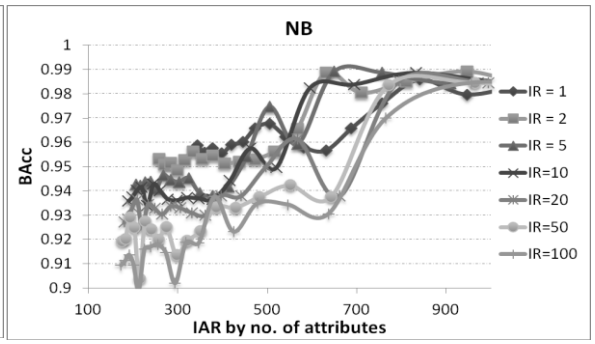
(c)



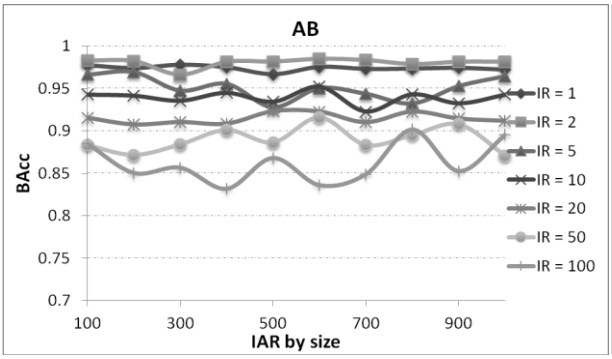
(d)



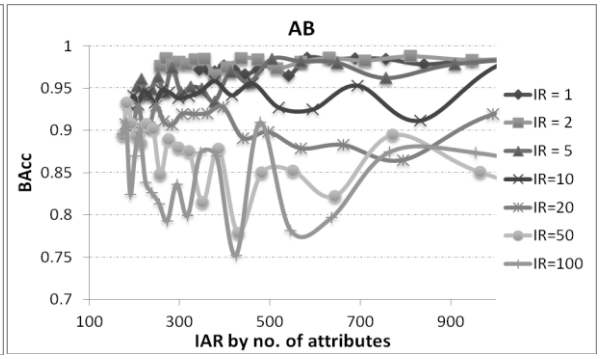
(e)



(f)



(g)



(h)

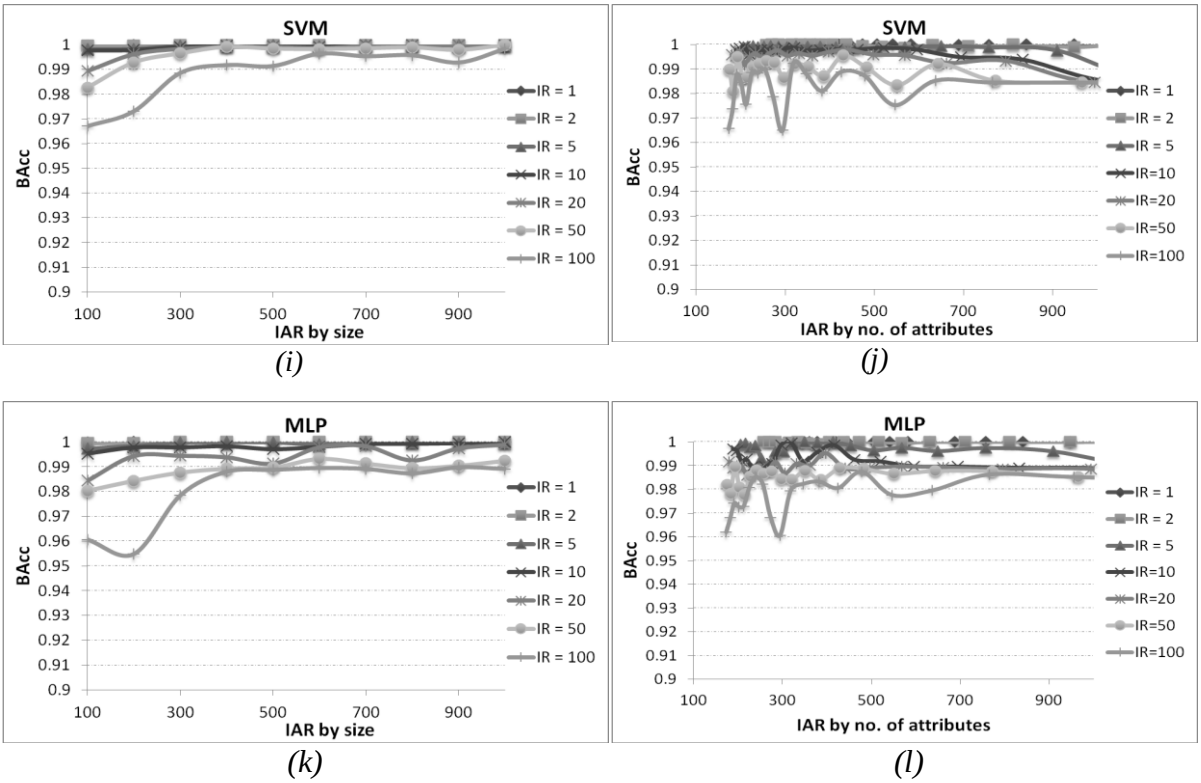


Figure 7.13 - The effect of varying IR and IAR on the performance of different classifiers

As it can be observed from the diagrams, the results obtained on the same classifier in the two scenarios are similar, with the observation that the second scenario presents ampler variations. This is expected since removing one predictive attribute from the training set can produce more acute changes in performance than removing a subset of instances, if the size is reasonably large (in this situation, the smallest training set size reached was around 2200 instances). The trends of the curves obtained for the same classifier via the two scenarios are, however, similar.

The results indicate that, generally, for the same IR, the performance improves as IAR increases (as expected). Another observation is related to the fact that as the IR increases, better performance is achieved at higher IAR values. The one exception is presented by AB: the curves for different IRs do not present an increasing trend. However, as IR increases, the instability of AB is more pronounced (the variations between different IAR values become more ample). This inconsistent behavior was observed for AB in the earlier evaluations as well. Also, AB seems to be affected the most by the imbalance – if, at IR = 1, its BAcc values are around 0.98, when IR = 100, they decrease below 0.85. A somewhat unexpectedly good behavior is observed for SVM – high BAcc values even at high IR values and stable across different IAR levels. As before, this is the result of the existence of the appropriate support vectors in the training data. As expected, the MLP yields good performance and increased stability with respect to IR and IAR variations – its BAcc values never decrease below 0.96, even at high IR and small IAR values.

To conclude, this experimental study has indicated that all methods are affected by the imbalance. Decision trees are greatly affected when the data is imbalanced, but reducing the level of pruning improves their performance considerably. As the IR increases, pruning deteriorates the performance of the decision tree model. This result supports the statement in [Wei04], that pruning might eliminate rare and important cases, thus affecting the correct identification of the minority class. However, no pruning at all results in an increase of complexity for the majority class as well, which might lead to over-fitting in that area. A more sophisticated approach is therefore required for imbalanced domains, an intelligent

pruning mechanism, which adjusts the level of pruning for branches according to the number of minority cases they contain.

As opposed to the conclusions stated in [Jap02, Vis05], we found that SVMs are strongly affected by the imbalance problem. A justification for this difference could be found in the data employed for evaluation: in the case of artificial data (used by in [Jap02]), even for large IRs, the examples which represent reliable support vectors are present in the data, due to the systematic data generation process (and, hence, the data periodicity), while in the case of real problems (i.e. the benchmark data used in our evaluations), these vital learning elements might be missing. Also, out of the methods we have evaluated, MLPs have proved to be the most robust to the imbalance problem.

The reduction in performance becomes more severe as the IR increases. However, for the same IR, larger IAR values are associated with improved classifier performance. Therefore, techniques for increasing the value of IAR (i.e. larger dataset size and/or smaller complexity) may lead to an improved behavior.

Therefore, developing new, general methods to improve the robustness of traditional learning algorithms in imbalanced scenarios is necessary. In section 7.3 I will present a new general methodology as a solution for imbalanced classification problems.

7.2 State of the Art in Imbalanced Classification

Several different strategies for improving the behavior of classifiers in imbalanced domains have been reported in the scientific community. Broadly, the approaches for dealing with imbalanced problems can be split into: data-centered (sampling methods), algorithm-centered and hybrid solutions.

7.2.1 Sampling Methods

Sampling techniques focus on altering the distribution of the training data: either randomly, or by making an informed decision on which instances to eliminate or add (by multiplying existing examples, or artificially generating new cases). Under this category we find random over- and under-sampling, or more elaborated approaches, such as:

a. *Synthetic Minority Over-sampling Technique* (SMOTE) [Cha02]: which synthetizes new, prototype minority samples, thus pushing the separation boundary further into the majority class; it can be combined with random under-sampling;
b. *Tomek links* [Tom76]: a Tomek link is formed by 2 neighboring instances x_i and x_j belonging to *different* classes, if $\neg\exists\ x_l$ s.t. $d(x_i, x_l) < d(x_i, x_j)$ or $d(x_j, x_l) < d(x_i, x_j)$. According to the method, the two instances are either noise or borderline. Consequently, Tomek links can be employed both for sampling (by removing the majority class instances) and as a cleaning strategy (by removing both instances);
c. The *Condensed Nearest Neighbor Rule* (CNN) [Har68]: attempts to form a consistent subset of instances by removing majority instances which are distant from the decision border. The consistency is checked using a 1-nearest neighbor classifier (1-NN), i.e. a subset is consistent if using 1-NN all instances are correctly classified;
d. *One-Sided Selection* (OSS) [Kub97]: eliminates "unsafe" instances by applying first Tomek links as an under-sampling method (i.e. remove borderline/noisy majority examples), followed by the application of CNN (i.e. remove majority examples which are distant from the decision border);
e. The *Neighborhood Cleaning Rule* (NCL) [Lau01]: for each instance x_i find its three nearest neighbors; if x_i is misclassified by the neighbors and x_i belongs to the majority class, then x_i is removed; if x_i is misclassified by the neighbors and it

belongs to the minority class, then, out of the three neighbors, the ones belonging to the majority class are removed;

f. *Class Purity Maximization*(CPM) [Yoo05]: employs a hierarchical clustering technique to partition the data, until no reduction in cluster impurity can be found. The impurity is defined as the proportion of minority instances in the cluster.
g. *Under-Sampling Based on Clustering* (SBC) [Yen06]: initially clusters all instances in the dataset into k clusters. Then, it computes, for each cluster, the appropriate sample size for the majority class instances, given the overall IR and the cluster data. In each cluster, random under-sampling on the majority class is then applied.
h. *Evolutionary Under-Sampling* (EUS) [Gar09]: is an under-sampling method in which the search for the best sample is guided through evolutionary mechanisms; the fitness functions employed by the authors attempt to provide the optimal trade-off between balance in the distribution of classes and performance

Sampling methods can be employed as pre-processing techniques [Gar09]. This is both a blessing and a curse: a blessing because the computational effort to prepare the data is needed only once; a curse because it cannot be employed as a systematic method since there are no guidelines on which specific method is expected to produce the best quality dataset. In order to maximize the classification performance in the mining step, one should carefully match the appropriate sampling technique to the learning algorithm employed at that stage. For example, Support Vector Machines (SVM) should perform better when paired with a sampling strategy which cleans the boundary region, such as CNN or OSS, whereas the k-Nearest Neighbor may achieve better results with a neighborhood cleaning rule (NCL).

Also, some methods require the analyst to set the amount of re-sampling needed, and this is not always easy to establish. It is acknowledged that the naturally occurring distribution is not always the best for learning [Wei03]. A balanced class distribution may yield satisfactory results, but is not always optimal either. The optimal class distribution is highly dependent on the particularities of the data at hand. Moreover, as the dimension of the training set decreases, more positive examples are needed to induce a good model.

7.2.2 Algorithm-based Methods

Algorithm-centered techniques, also known as internal approaches, refer to strategies which adapt the inductive bias of classifiers, or newly proposed methods for tackling the imbalance. For decision trees, such strategies include adjusting the decision threshold at leaf nodes [Qui91], adapting the attribute selection criterion [Liu10], or changing the pruning strategy [Zad01, Lem11b]. For classification rule learners, using a strength multiplier or different algorithms for learning the rule set for the minority class is proposed in [Grz05], while for association rule learners, multiple minimum supports are employed in rule generation [Liu00]. In [Liu11], confidence weights are associated to attribute values (given a class label) in a kNN approach. For SVMs, class boundary alignment is proposed in [Wu03] and the use of separate penalty coefficients for different classes is investigated in [Lin02]. Newly proposed methods, which deal with the imbalance intrinsically, include the biased minimax probability machine (BMPM) [Hua06], or the infinitely imbalanced logistic regression (IILR) [Wil09].

7.2.3 Hybrid Methods

Hybrid approaches combine data- and algorithm-centered strategies. A number of approaches in this category consist of ensembles built via boosting, which also employ replication on minority class instances to second the weight update mechanism, in the attempt to focus on the hard examples. Also, the base classifiers may be modified to tackle

imbalanced data. Such approaches include SMOTEBoost [Cha03], DataBoost-IM [Guo04], and a complex SVM ensemble [Tia11].

Another hybrid strategy which may prove beneficial in imbalanced problems is the one employed in cost-sensitive problems, to bias the learning process according to the different costs of the errors involved [Dom99, Zho06, Sun07].

Two main directions for cost-sensitive methods employed in imbalanced classification have been identified:

- Consider the cost matrix known [Tin02, Liu06]
- Utilize a cost matrix which compensates for the value of the IR [Mar00, Han06]

Unfortunately, the cost matrix is seldom known in real world problems, and this is one of the open issues in cost sensitive learning – employing an appropriate cost matrix. Also, cost-sensitive learning by IR compensation is inappropriate for the following reason: extensive empirical evaluations performed in [Wei03] show that the best distribution for learning is not the balanced distribution, but depends on the problem at hand.

The strategy we propose in this paper addresses the above mentioned drawbacks, by identifying the best cost matrix for a given problem via evolutionary search strategies. The search criterion, i.e. the fitness function of the genetic algorithm, can be specified according to the particularities of the given problem. Selecting the appropriate fitness criterion is in closer relation to specific domain goals, than setting the exact costs in the cost matrix.

7.3 ECSB: Evolutionary Cost-Sensitive Balancing

Imbalanced class distributions are common in real world data. Our analyses have shown that the performance of all classifiers is affected under such conditions. Out of the existing solutions, sampling methods can be employed as pre-processing strategies; however, some techniques require experience for applying them properly; moreover, to maximize their effect, they should be matched with the learning method – again – requiring experience. Modifications to basic algorithms have also been proposed in the literature, with good performance improvements, but each is restricted to a specific class of techniques. To address these issues, we propose a new general methodology for classification in imbalanced domains: *Evolutionary Cost-Sensitive Balancing* (ECSB). The objective of the ECSB method is to improve the performance of a classifier in imbalanced domains. It is a meta-methodology, which can be applied to any error-reduction classifier. Two strategies are simultaneously followed by the method: (1) use a cost-sensitive meta-classifier to adapt to the imbalance and (2) tune the base classifier's parameters.

7.3.1 Method Description

The outcome of the method is a tuple <M, S> for the triple <p, i, m>, where M is a cost matrix and S is the set of resulting parameter settings for the given problem – data (d), selected classifier (c) and performance metric (p). M is employed in conjunction with the cost-sensitive classifier, in order to build a more efficient classification model, focused on better identifying the underrepresented/interest cases. The search for M and S is performed through evolutionary mechanisms. The cost-sensitive component employs a meta-classifier to make its base classifier cost-sensitive, taking into account the misclassification costs. The main mechanisms for wrapping cost-sensitivity around traditional classifiers usually focus on employing a larger penalty for the errors on classes with higher misclassification cost, or modifying the training data such that the costly cases are proportionally better represented than the others.

The general flow of the method is presented in Figure 7.14. The inputs are: the problem (d), translated in terms of a set of labeled examples (i.e. the training set), the base classifier (c) and the metric (p) to use for assessing the performance of *c*.

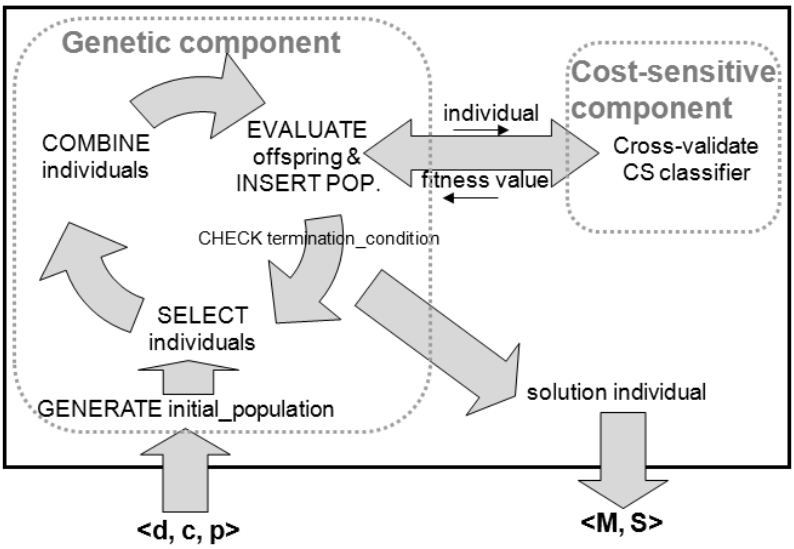


Figure 7.14 – General ECSB flow

The result of the method is a <M, S> tuple, which is used by a (meta-) cost-sensitive classifier to build the final classification model.

The Cost-Sensitive Component

A discussion on the types of costs and related definitions has been presented in Chapter 6. For the purpose of imbalanced classification, the focus is on misclassification costs alone, since they can be employed to bias the learning process such as to provide a better identification for the minority class instances. As presented previously, misclassification costs are represented via a cost matrix $M = (c_{ij})_{nxn}$. One of the most important difficulties when dealing with different error costs is the quantification of misclassification costs. Even if it is relatively easy to determine which errors are more severe than others (e.g. in medical diagnosis $c_{12} > c_{21}$), it is difficult to quantify the gravity of an error exactly, since this may translate, indirectly, into more serious social/moral dilemmas, such a putting a price tag on human life.

In the ECSB approach, the cost matrix (M) for the given imbalanced problem is determined indirectly, following a genetic search. The result of the search is influenced by tuning the fitness function employed, which can be more easily translated, given a specific problem, than directly setting the cost matrix. For example, it is more reasonable to state that the objective is to maximize both TP_{rate} and TN_{rate} in medical diagnosis, or to maximize precision in online advertising, than it is to set specific error costs.

The implementation of the cost-sensitive component considers three cost-sensitive strategies:

(1) Reweight/resample training instances according to the total cost assigned to each class (*CSr*) [Wit05];

(2) predict the class with minimum expected misclassification cost, instead of the most likely class (*CS*) [Wit05]:

$$prediction\ (x) = \arg\min_i L(x,i) \tag{7.3}$$

$$L(x,i) = \sum_j P(j \mid x) c_{ij} \tag{7.4}$$

(3) Utilize an ensemble method to re-label the training instances according to the Bayes optimal prediction principle, which minimizes the conditional risk (*MC*) [Dom99].

The Genetic Component

We have utilized the General Genetic Algorithm Tool for implementing the genetic component [Der02]. It provides the traditional genetic algorithms (GA) search organization, parent selection and recombination techniques. The specificity of our implementation is the problem representation and the fitness function(s) employed. The following sub-sections present the GA flow and the employed GA mechanisms, and the specific problem representation.

Search Organization

The search process starts with the initial population, i.e. a set of potential solutions, generated randomly (lines 1 and 2 in the pseudocode snippet below). By repeatedly applying recombination operators to some of the individuals in the population over a number of cycles, an element (or group of elements) is expected to emerge as a good quality approximate solution to the given problem (the loop between lines 3 and 9). Following a strategy similar to steady state evolution, in each cycle a number of new offspring is generated (additional pool). After evaluating their fitness (line 7), the fittest *p_size* individuals out of the old population and the additional pool (the newly generated offspring) will constitute the new population (line 8):

```
(1)   population = generate_initial_population(p_size)
(2)   evaluate_fitness (population)
(3)   repeat
(4)         parents = select(population)
(5)         offspring = crossover(parents)
(6)         mutate(offspring)
(7)         evaluate_fitness (offspring)
(8)         insert (offspring, population)
(9)   until (termination_condition)
(10)  return best_individual
```

This strategy considers elitism implicitly. The search process stops when one of the following occurs: the optimal fitness value is reached, the difference between the fitness values of the best and the worst individuals in the current population is 0, or a fixed (pre-determined) number of crossover cycles have been performed:

Representation and Fitness Function

Each individual consists of four chromosomes (Figure 7.15): the first two representing each a misclassification cost (elements of M), and the last two representing parameters for the base classifier (elements of S). Although we have considered only two parameters for S – since most base classifiers used in the experiments have only two important learning parameters – the method can be extended to search for a larger number of parameters, depending on the tuned classifier. The first two chromosomes in the individual represent the meaningful coefficients of the 2x2 cost matrix. We assume the same reward (i.e. zero cost) for the correct classification of both minority and majority classes. Each chromosome consists of 7 genes, meaning that each cost is an integer between 0 and 127. We considered this to be sufficient to account even for large IRs. Gray coding is employed to ensure that similar genotypes produce close manifestations (phenotypes).

Fitness ranking is used to avoid premature convergence to a local optimum, which can occur if in the initial pool some individuals dominate, having a significantly better fitness than the others. Since establishing how to assess performance is essential in imbalanced problems and there is no universally best metric, which captures efficiently any problem's goals, we have implemented several different fitness functions, both balanced and (possibly) imbalanced.

$c_{1,2}$	$c_{2,1}$	$setting\_value_1$	$setting\_value_2$

Figure 7.15 – Individual representation

For consistency with the literature, we sometimes employ TP_{rate} and sometimes recall for referring to the same measure:

1. **GM** (geometric mean) = $\sqrt{TP_{rate} * TN_{rate}}$ (7.5)
2. **BAcc** (balanced accuracy) = $\frac{TP_{rate} + TN_{rate}}{2}$ (7.6)
3. **FM** (f_β-measure) = $(1+\beta^2)\frac{prec * recall}{prec + recall}$ (7.7)
4. **LIN** (linear combination between TP_{rate}, TN_{rate}) = $\alpha * TP_{rate} + (1-\alpha) * TN_{rate}$ (7.8)
5. **PLIN** (linear combination between recall, prec.) = $\alpha * Recall + (1-\alpha) * Prec$ (7.9)

7.3.2 Experimental Evaluation

This section presents the experiments performed to validate the ECSB method and to compare it with recent proficient strategies. The next sub-section presents the general setup: it includes the evaluation methodology employed throughout the experiments, as well as the mechanisms and settings employed. Three different evaluation suites are then presented, with discussions of the results. A first set of tests evaluates comparatively the performance of different specializations of ECSB on large IR, small IAR datasets, since previous analyses [Lem11b] have shown that classifiers are most affected on such problems; the second presents a comparison between ECSB and a prominent under-sampling strategy for imbalanced data: Evolutionary Under-Sampling [Gar09]; and the third analyzes the improvement of ECSB on the SVM classifier, in comparison with a recent SVM ensemble method for imbalanced problems [Tia11].

Experimental Setup

Experiments have been carried out following a 2-fold cross-validation schematic (except for the third set of experiments). Generally, the following have been compared: (1) the results of the classifier on the imbalanced domain with default settings (*Base*) with (2) the results obtained by the same classifier following data pre-processing with SMOTE [Cha02] and default settings (*Base+SMOTE*), (3) the results obtained by the classifier on the imbalanced domain following a parameter tuning stage, performed with the genetic component of ECSB (*ECSBT*) and (4) the results obtained by a classifier wrapped in our ECSB method (*ECSB*).

The specific mechanisms and setting values employed for the genetic component are presented in Table 7.5. Several fitness functions have been considered. No tuning has been performed on the settings of the component so far. Five classifiers have been included in the experimental study, belonging to different categories: lazy methods (k-nearest neighbor – kNN), Bayesian methods (Naïve Bayes – NB), decision trees (C4.5), support vector machines (SVM) and ensemble methods (AdaBoost.M1 – AB). MLP has been excluded from these experiments because it generally proved to be more robust than the other five methods in imbalanced scenarios, and therefore the necessity for improvement is not as acute; moreover, it is impractically slow in combination with the ECSB method. Table 7.6 describes the parameters considered for each classifier (for ECSB and ECSBT).

Table 7.5 – Specific genetic mechanisms employed

Setting	Value
Population type	Single, similar to steady state
Initial population generation	Random
Population size	20
Additional pool	10
Crossover cycles	200
Parent Selection	Roulette wheel
Recombination Operators	Crossover: random crossover, 4 points Mutation: single bit uniform mutation, 0.2 rate
Fitness functions	GM; BAcc; FM; LIN; PLIN
Other	Fitness ranking Elitism, implicit with use of single population

Table 7.6 – Classifier parameters considered

Classifier	Parameters	Type and range
kNN	K – number of neighbors	Integer between 1 and 10
C4.5	C – confidence ratio	Real, between 0 and 0.4
	M – minimum number of instances per leaf	Integer, between 1 and 5
NB	n.a.	n.a.
AB	P – weight threshold for weight pruning	Integer, between 1 and 127
	I – number of iterations	Integer, between 1 and 30
SVM	C – complexity	Real, between 1 and 100
	E – exponent	Integer, between 1 and 11

General validation on large IR, small IAR datasets

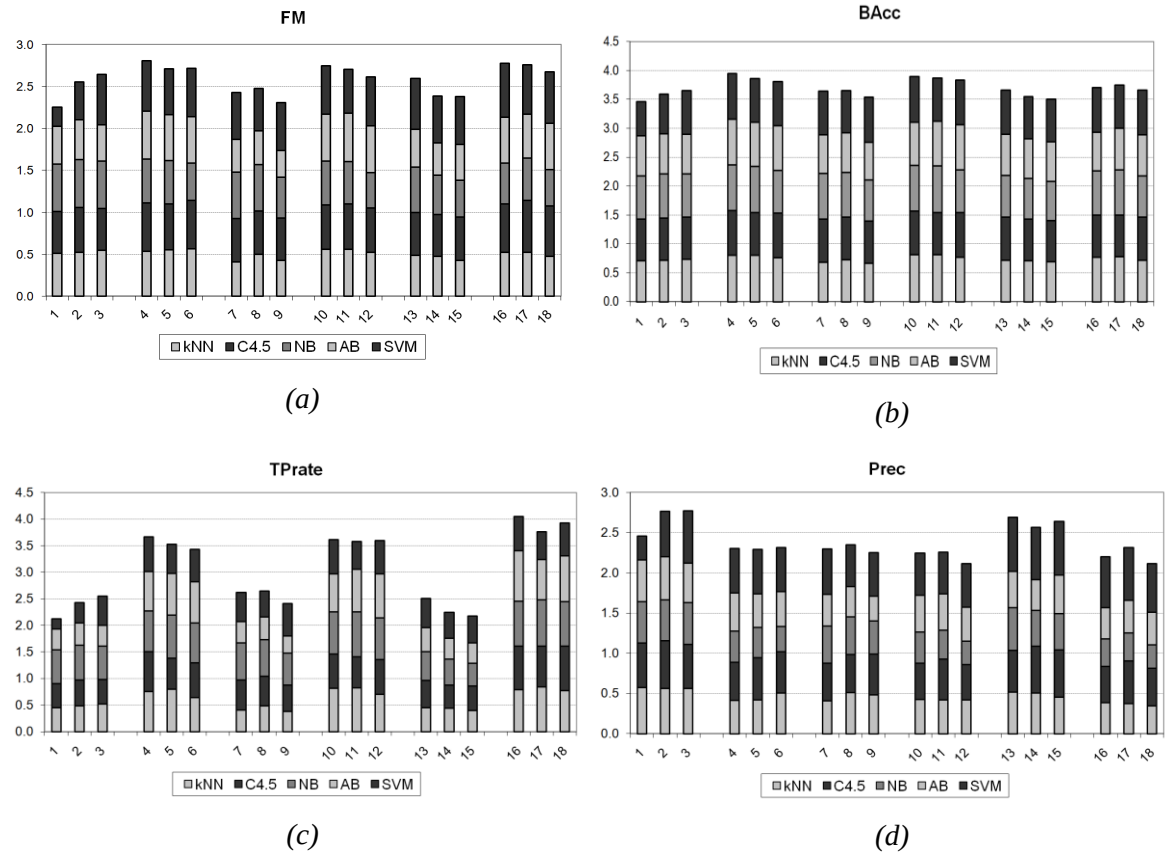
A first analysis has been performed on benchmark datasets having large IR and small IAR, as considered in [Lem11b], i.e. five datasets with IR between 5 and 16 and IAR below 60 (Table A.7.2, Appendix A). This combination of imbalance-related factors has been shown to produce a strong reduction in the performance of classifiers. Our experiments have yielded an average TP_{rate} value between .27 (SVM) and around .6 (NB and MLP). All three cost-sensitive strategies were considered (MC, CS and CSr), and five different fitness functions (GM, BAcc, FM with $\beta=1$, LIN and PLIN, the last two having $\alpha=0.7$). This results in 15 combinations for the ECSB method, compared with the results obtained by the classifier alone (*Base*), the classifier with SMOTE (Base+*SMOTE*) and the classifier with tuned parameter values (*ECSBT*).

The results are presented in fig. 7.16. For viewing purposes, the different methods have been numbered from 1 to 18; please refer to the legend for identification. Each bar in the diagrams represents the overall average score (under the specific metric) obtained by all five classifiers, using the corresponding method. For example – in diagram (c), the first bar represents the overall average TP_{rate} obtained by all five classifiers on all datasets, under imbalance conditions (~2.2), while the fourth bar represents the overall average TP_{rate} obtained by all five classifiers on all datasets obtained by ECSB using BAcc as fitness measure and CS as cost-sensitive strategy (~2.8).

Several remarks can be made regarding these results: (1) using balanced metrics as fitness measures, such as GM or BAcc, produces significant improvements in the TP_{rate} (second and fourth groups in Figure 7.16 (c)) and good improvements in FM and BAcc (second and fourth groups in 7.16 (a) and (b)); (2) FM is not effective as fitness measure (third group in all diagrams); (3) the linear combination between TP_{rate} and TN_{rate} ($\alpha=0.7$) as

fitness function does not improve TP_{rate} significantly (fifth group in 3.c), but instead it improves Prec (fifth group in 7.16.(d)); (4) the linear combination between recall and precision (α =0.7) as fitness score yields the most important improvement in TP_{rate} (last group in 7.16.(c)), but it degrades precision (1.16.(d)) – since α=0.7, more importance is given to improving recall than to precision; (5) for the SVM, both the TP_{rate} and the precision are significantly improved through the ECSB method (7.16.(c) and (d), the top portion of the bars); (6) out of the three cost-sensitive strategies evaluated, the most successful is CS (the first bar in each group from the second to the last), i.e. predict the class with minimum expected misclassification cost, instead of the most likely class.

Therefore, balanced metrics (except FM) are generally appropriate as fitness measures for ECSB in imbalanced problems; when the recall is of utmost importance (e.g. medical diagnosis), using the linear combination between recall and precision, with a high value for α, is appropriate; this is also suitable when both precision and recall (TP_{rate}) are important (e.g. credit risk assessment), but with a lower value for α. Cost-sensitive prediction is the most appropriate strategy to employ.



1 – Base
2 – Base+SMOTE
3 – ECSBT
4 – ECSB(CS, BAcc)
5 – ECSB(CSr, BAcc)
6 – ECSB(MC, BAcc)
7 – ECSB(CS, FM)
8 – ECSB(CSr, FM)
9 – ECSB(MC, FM)
10 – ECSB(CS, GM)
11 – ECSB(CSr, GM)
12 – ECSB(MC, GM)
13 – ECSB(CS, LIN)
14 – ECSB(CSr, LIN)
15 – ECSB(MC, LIN)
16 – ECSB(CS, PLIN)
17 – ECSB(CSr, PLIN)
18 – ECSB(MC, PLIN)

Figure 7.16 – F-measure, Balanced accuracy, TPrate and Precision obtained by the various methods on the large IR, small IAR data

Comparative Analysis with Evolutionary Under-Sampling

A second analysis was performed on a set of 28 imbalanced benchmark problems (Table A.7.3, Appendix A) from [Gar09], in order to compare our results with the performance of the Evolutionary Under-Sampling (EUS) strategy presented there. EUS has been shown to produce superior results when compared to state-of-the-art under-sampling methods, making it a good candidate for imbalanced datasets, especially with a high imbalance ratio among the classes. In this set of experiments, we have employed CS as cost-sensitive strategy and GM as fitness function – because it is the function employed in the most successful EUS model. We have also considered in the comparison the classifier with default settings (*Base*), the classifier with SMOTE and default settings (Base+*SMOTE*) and the classifier with tuned parameter values (*ECSBT*).

The results of this second analysis are shown in Tables 7.7 and 7.8. It can be observed that ECSB significantly boosts the performance of classifiers when compared to their behavior on the original problem (except for the AUC for AdaBoost.M1 – Table 7.12); on the average, there is ~25% relative improvement on the GM and ~5% on the AUC; the most significant improvements have been obtained for the SVM classifier (~ 86% relative improvement on GM and 16% on AUC). Also, it yields significant improvements over SMOTE and ECSBT (~17% and ~14%, respectively, relative improvement on GM and ~5% and ~2%, respectively, on AUC). Slight improvements over the best EUS method have also been observed (i.e. the specialization of EUS which achieved the best performance in the above cited work): up to 9% relative improvement in AUC.

Table 7.7 – Average GM (with standard deviations) obtained by the various methods

GM	**Best EUS [Gar09]**		**Base**		**Base +SMOTE**		**ECSBT**		**ECSB**	
	mean	*stddev*	*mean*	*stddev*	*mean*	*stddev*	*mean*	*stddev*	*mean*	*stddev*
kNN	.797	*.169*	.731	*.225*	.744	*.218*	.762	*.230*	.817	*.173*
C4.5			.660	*.317*	.716	*.254*	.635	*.307*	.796	*.179*
NB			.754	*.202*	.771	*.164*	.754	*.202*	.814	*.129*
AB			.640	*.314*	.658	*.306*	.619	*.323*	.798	*.188*
SVM			.431	*.401*	.558	*.358*	.750	*.213*	.803	*.184*

Table 7.8 – Average AUC (with standard deviations) obtained by the various methods

AUC	**Best EUS [Gar09]**		**Base**		**Base +SMOTE**		**ECSBT**		**ECSB**	
	mean	*stddev*	*mean*	*stddev*	*mean*	*stddev*	*mean*	*stddev*	*mean*	*stddev*
kNN	.809	*.170*	.803	*.144*	.803	*.144*	.848	*.140*	.867	*.128*
C4.5			.797	*.147*	.797	*.147*	.786	*.157*	.830	*.125*
NB			.873	*.110*	.873	*.110*	.874	*.111*	.874	*.111*
AB			.892	*.105*	.892	*.105*	.891	*.098*	.878	*.121*
SVM			.714	*.175*	.714	*.175*	.790	*.143*	.830	*.132*

Comparison with a SVM Ensemble Method

To further validate our method, experiments on several datasets reported in [Tia11] have been conducted, to compare the ECSB method with the complex SVM ensemble method proposed there. Its effectiveness has been shown through comparisons with other available solutions: sampling (under- and over-) and ensemble approaches (bagging and boosting), under various metrics: precision, recall and f-measure. The reason for performing such an analysis can be found in [Lem11b], where it has been discovered that the performance of the SVM is significantly reduced in imbalanced domains.

Table 7.9 – Recall, precision and f-measure obtained by ECSB, compared to the SVM ensemble method

	ECSB		**SVMEns [Tia11]**	
	mean	*stderr (mean)*	*mean*	*stderr (mean)*
Recall				
Breast-cancer	**.513**	*.062*	.509	*.011*
Cars	**1.0**	*.0*	.977	*.005*
Glass	**.8**	*.07*	.65	*.017*
Balance-scale	**.92**	*.042*	.879	*.022*
Average	**.808**	*.044*	.753	*.01*
Precision				
Breast-cancer	.453	*.033*	**.475**	*.009*
Cars	**1.0**	*.0*	.124	*.004*
Glass	.909	*.045*	**.929**	*.005*
Balance-scale	.082	*.277*	**.140**	*.015*
Average	**0.611**	*.089*	.417	*.008*
F-measure				
Breast-cancer	.457	*.043*	**.491**	*.006*
Cars	**1.0**	*.0*	.213	*.005*
Glass	**.822**	*.048*	.764	*.008*
Balance-scale	**.486**	*.027*	.241	*.011*
Average	**0.691**	*.03*	.427	*.008*

Ten-fold cross-validation was employed in these experiments; for the ECSB method, BAcc was used as fitness function and CS as cost-sensitive strategy. Due to time limitations, the analysis has been restricted to the first four datasets employed in [Tia11] – Table A.7.4, Appendix A.

The results are presented in Table 7.9. They indicate that the ECSB method achieves more significant improvements than the SVM ensemble method in terms of recall, keeping precision at approximately the same levels (in three out of the four datasets, the F-measure has significantly higher values for ECSB than for SVMEns). On the average, the relative improvement on recall is of ~7%, and on FM of ~60%.

7.4 Conclusions on Imbalanced Classification

All traditional algorithms are affected to some extent by the class imbalance problem. Also, the correct choice of the metric (or combination of metrics) to assess – and ultimately improve, is essential for the success of a data mining effort in such areas, since most of the time improving one metric degrades others.

A series of methods which deal with the class imbalance have been proposed in the literature over the last years. Sampling strategies are important because they can be used as pre-processing strategies. However, some approaches are difficult to employ by a less experienced user – e.g. some require setting the amount of sampling. Most importantly, to maximize their effect, they need to be matched to the specific classifier employed. Modifications to basic algorithms have also been proposed in the literature, with good performance improvements, but each is restricted to a specific class of techniques.

A first original contribution presented in this chapter is the systematic study which assesses the behavior of traditional classification algorithms under imbalanced class distributions. A large number of real-world benchmark datasets have been considered, of different sizes, IR, IAR and complexities. Representative algorithms belonging to a wide

spectrum of techniques have been included in the study and various performance metrics have been measured.

The results have confirmed that all methods suffer, to different extents, of performance degradation in such scenarios, with the MLP being – in general – the most robust, and the SVM the most prone to performance degradation. Also, the IAR, which encapsulates size and complexity information, provides a better characterization of a dataset than the size and complexity measures taken separately. The IAR meta-feature also represents an original contribution. Reducing the level of pruning improves the decision trees' capacity to identify minority class instances.

To overcome the above mentioned limitations, a new general hybrid strategy for improving the performance of classifiers in imbalanced problems has been proposed. The method, Evolutionary Cost-Sensitive Balancing (ECSB), is a meta-approach, which can be employed with any error-reduction classifier. Two strategies are followed by the method simultaneously: tune the base classifier's parameters and use a cost-sensitive meta-classifier to adapt to the imbalance. A great advantage of the method, besides its generality, is that it needs little knowledge of the base classifier; instead, it requires specific knowledge of the domain to select the appropriate fitness measure.

We have performed several evaluations on benchmark data, to validate the method and compare it with current state of the art strategies for imbalanced classification. The results have demonstrated the following:

- the ECSB method significantly improves the performance of the base classifiers in imbalanced conditions, achieving superior results to sampling with SMOTE or adapting the algorithm to the imbalance via evolutionary parameter selection;
- ECSB achieves superior results to current prominent approaches in literature: Evolutionary Under-Sampling and a complex SVM ensemble;
- the most successful cost-sensitive strategy is predicting the class with minimum expected misclassification cost, instead of the most likely class (CS);
- balanced metrics are generally appropriate as fitness functions (except for the F-measure); for extreme problems – e.g. precision is of utmost importance, or recall is the only important measure – imbalanced metrics, such as the parameterized linear combination of recall and precision (with the appropriate value given to α) are more suitable.

Our current focus is on improving the method training time, which is influenced by the size of the data and the base classifier employed. At the moment we have experimented with a sequential implementation, but the method presents a great parallelization potential and we expect that the parallel version will run significantly faster. Also, in the current implementations we have experienced with a fixed set of GA parameters, which cannot be the best for all problems. Adding an extra layer to the genetic search component, which will focus on finding the most suitable GA parameters for the given problem, is also a current focus.

The original ECSB method, the new data meta-feature (IAR) and the associated experimental studies presented in this chapter have been disseminated through a research paper submitted at the *Data Mining and Knowledge Discovery Journal* (revision requested):

1. Potolea R. and **Lemnaru C.**, "Evolutionary Cost-Sensitive Balancing: A Generic Method for Imbalanced Classification Problems", submitted at *Data Mining and Knowledge Discovery*, revision requested

a research paper in *Lecture Notes in Business Intelligence Processing* (in publication):

1. **Lemnaru C**. and Potolea R., "Imbalanced Classification Problems: Systematic Study, Issues and Best Practices", to appear in *Lecture Notes in Business Information Processing*, 2012

and 2 research papers, presented at renowned international conferences:

1. Potolea, R., **Lemnaru, C**., "Dealing with Imbalanced Problems: Issues and Best Practices", *Proceedings of the 12th International Conference on Enterprise Information Systems*, Volume 2, AIDSS, June 8 - 12, pp. 443-446, ISBN 978-989-8425-05-8, 2010
2. Potolea, R., **Lemnaru C.**, "A Comprehensive Study of the Effect of Class Imbalance on the Performance of Classifiers" *Proceedings of the 13th International Conference on Enterprise Information Systems*, vol. DISI, pp. 14-21, 2011

8 Case Studies

This chapter presents a series of case studies, which follow the general DM process in the attempt to tackle certain problems which arise in specific application domains, by providing "recipes" with a certain level of generality, such that they are not restricted to the exact problem constraints they are designed for. All solutions investigate several DM steps and some apply methods from the previous chapters. The problems tackled are: automatic classifier selection via meta-learning, which resulted in a framework that can be used for classifier selection in any problem domain; data partitioning strategies, which have been analyzed both generally and in the specific context of digital signature recognition, resulting in a hierarchical model for offline signature recognition; speed-up through parallelization, which resulted in a parallel implementation of a well-known decision tree classifier and a parallel GPGPU-based lightweight genetic framework; specific real-world DM application scenarios – community structure detection in social mining, spam prediction in spam filtering, user type identification in adaptive e-learning systems and a semi-supervised opinion mining technique.

8.1 Meta-learning: Automated Classifier Selection

Selecting the appropriate learning algorithm for a new problem, given its specific particularities, is a complex task, even for the experienced data analyst, as hidden knowledge is present in data. Such knowledge can seldom be surmised by the domain experts, and almost never by the data analyst. Therefore, an initial assessment should be performed, in order to identify the most promising knowledge extraction methodology for the given problem. The process usually involves creating several models with different learning algorithms under various settings and evaluating their performance with respect to the requirements of the problem. The analyst can then choose the learning algorithm and the settings which best fit the context. The time required to build a model increases with the complexity of the model and with the size of the input data. Running and evaluating a large number of learning algorithms is therefore unfeasible.

A suitable approach involves comparing the new problem with a set of problems for which the learning algorithm performance is already known [Ben00, Vil04]. The analyst must identify the problem which most resembles the analyzed data. Consequently, the same learning algorithm and settings which obtained the best results on the former problem(s) is expected to achieve similar performance on the new problem. To make use of this approach, the expert should evaluate with various techniques a large amount of problems. Also, the success of the selected learning algorithm on the new problem depends on the expert's strategy for selecting similar problems.

This section offers an overview on the efforts we have invested into developing a semi-automated classifier selection framework which employs meta-learning techniques to indicate the most appropriate prediction strategies for a new data mining problem.

8.1.1 Baseline Performance Assessment

Classification algorithms are different in their approach, and hence they achieve different performance levels for different applications. These particularities have led to an increasing interest towards trying to combine the predictions of several algorithms, in order to obtain a scheme that performs well in several different areas. Ensemble methods are known to reduce the variance of learning algorithms, while the bias remains unchanged, or increases. This means that if the base classifier possesses a large bias on a given problem (i.e. the decision boundary doesn't match the form of the separation boundaries the algorithm can learn), the ensemble will also possess a large bias, resulting in a low performance. In addition, establishing a lower threshold to the accuracy on a certain problem is essential in classifier

selection, for establishing the baseline performance. Thus, we have developed a classifier fusion system based on the principles of the Dempster-Shafer theory of evidence combination [Mol10, Mur10]. The system tackles the advantages of combining different sources of information to reduce bias and attain a high degree of stability across different problem domains. The uncertainty evaluation provided by the Dempster-Shafer theory also contributes to achieving this stability. Comparative evaluations have been performed on benchmark data, using representative algorithms from various categories [Mur10]. The results have indicated that the best Dempster Shafer combination of classifiers is always superior to the average of individual classifiers that it combines. More than that, the combination yields better performance than the best individual classifier on four of the eight datasets investigated. The second best combination almost always exceeds the average of individual classifiers. Both best and second best combinations are constantly (with a single exception) represented by 5-classifier pair-wise combinations. A combination which repeatedly yields high performance values is the ((NB, J4.8), (SVM, MLP), KNN) pair-wise combination. The superiority of (DT, BN) pair is expected, since the two classifiers are biased by different types of non-relevant features. The weakness of one learner (correlated features for NB, irrelevant for DT) is the strength of the other one, so they compensate each other. Also, while SVM may be seriously affected by the absence of appropriate support vectors (i.e. lack of important instances in the training set), MLP may compensate such a deficiency through weights adjustments in the training stage, and thus compensates the SVM weakness. The relative performance improvement in the best case, compared to the average performance of the individual classifiers is of up to 3.69, and 1.38 compared to the individual classifiers involved in the combinations.

Therefore, the evaluations have confirmed the assumptions related to stability and thus allowed the formulation of a method for establishing the baseline accuracy for any problem domain. The choice of a specific learning scheme for a certain problem is justified only if its performance is better than that of the system.

8.1.2 A Framework for Automated Classifier Selection

In [Cac09, Cac10, Pot11b], we have designed and evaluated an evolutional meta-learning framework for automatic classifier selection, which ranks and suggests accurate learning algorithms for a new problem submitted to the system (together with specific user requirements). The highest ranked algorithm is expected to induce a model which achieves the best performance under the given conditions. Several original design elements have been introduced into the framework. The user involvement is limited to providing the input problem (i.e. a dataset which the user needs to classify) and specifying the requirements (such as the interest metric, interpretable vs. non-interpretable model, etc). The result is presented as a list of learning algorithms, arranged in decreasing performance order. Recommending more than a single learning algorithm is important as it allows users to decide on the specific strategy they want to follow (from the trade-off between speed and performance, to the availability of tools, or existing knowledge to deal with necessary techniques). The system also minimizes the time it takes to provide the results by organizing tests in a queue. The queue is ordered according to priority and it pushes slower tests to the end. The system's knowledge about the dataset space continuously improves by storing the results of all the tests run on new datasets. As more datasets are stored in the database, new (closer) neighbors are introduced, thus increasing the accuracy of the predicted order of tests.

The conceptual model of the framework is presented in figure 8.1. The process of obtaining the predictions is roughly divided into selecting the similar problems and obtaining predictions from similar problems. In order to provide accurate predictions for new datasets the system relies on existing problems and the solutions obtained on those problems (classifier + performance). It must have the ability to increase its knowledge by adding new

problems and the corresponding solutions. The evaluation of the performance of the algorithms on existing problems is performed mainly in the initialization phase.

After a significant number of problems have been collected, the system is expected to produce reliable outcomes, in the prediction phase. In this phase, a new problem is submitted to the system, along with its requirements. The system must find similar problems in its collection. The similarity refers to the resemblance, in terms of meta-features, between the problem under evaluation and problems in the collection, and is evaluated by computing a distance between the analyzed problem and the stored problems. A subset of the nearest stored problems is selected as neighbors of the analyzed problem.

The results obtained by the learning algorithms for every neighbor problem are known (represent background knowledge, stored in the collection, together with the datasets and their meta-features). The performance score of each classifier is obtained by inspecting the results obtained by that classifier from the perspective of the user requirements. The framework then predicts the performance score for the classifier on the analyzed problem as a combination of the performance scores obtained by that classifier on the neighboring problems. The final list of recommended learning algorithms is ordered by their predicted performance scores. An extension of the initialization phase runs during the system idle time, when no predictions need to be performed. More specifically, the system extends its knowledge by evaluating models on newly added problems and saving the results.

The meta-features employed for dataset distance computation have been grouped into four categories: (1) general attribute-related meta-features (such as number of attributes and their types), (2) properties of the nominal and binary attributes (such as the minimum and the maximum number of distinct values, the mode and the standard deviation, the mean entropy), (3) the properties of the continuous attributes (mean skewness, mean kurtosis) and (4) dataset dimensionality (number of instances, imbalance rate).

Various performance metrics have been introduced within the framework, both balanced and imbalanced. For allowing users from different areas of expertise to discover a generally good classifier, an original evaluation metric has been proposed, which combines the accuracy, the geometric mean and area under the ROC curve:

$$gM = \frac{Acc + GM + AUC}{3} \tag{8.1}$$

The classifier prediction process is divided into three phases: distance computation, neighbor selection and prediction computation (or voting). For each of these phases we have proposed and evaluated several strategies [Pot11b]:

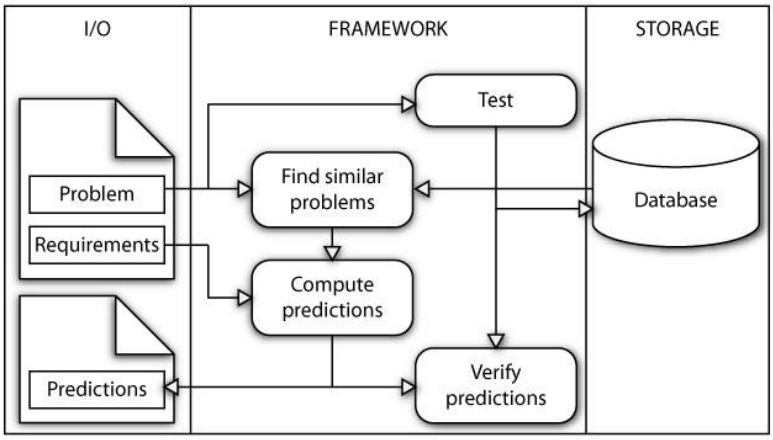


Figure 8.1 – Conceptual framework model

- Distance computation: Euclidean, Chebyshev, Top 3 Chebyshev, all normalized
- Neighbor selection: Top 3, Above Middle (compute the mean of the closest and furthest neighbors and use it as selection threshold), Above Median, Above Mean
- Voting strategy: Democratic, Weighted

The prediction process should produce performance estimates which are as close to the actual performance achieved by the classifier on the dataset, without surpassing it. Also, strategies which minimize the deviation means for all metrics considered should be preferred.

Experimental evaluations

The system has been initialized with 19 benchmark datasets that range from very small to medium sizes (up to 6000 instances). Also, the following classifiers are available: Bayes network, Naive Bayes, decision trees, neural network, support vector machines (using their implementations provided by WEKA). Evaluations have been performed using all the possible combinations of the implemented strategies for distance computation, neighbor selection and performance score prediction.

To perform a test suite, a performance metric has been selected and the following steps have been performed:

1. select a strategy combination
   a. select a dataset and use it as the analyzed dataset
      i. use the remaining 18 datasets as datasets stored in the system
      ii. use the selected strategy combination to predict performance
      iii. compare the predicted performance with the actual performance obtained in the initialization stage on the selected dataset
   b. select next dataset
2. compute the deviation mean and the absolute deviation mean on all datasets and classifiers for this strategy
3. select next strategy combination

The above strategy has been applied for the following metrics: accuracy, geometric mean, generalized geometric mean, area under ROC, general purpose metric. The deviation between the predicted and true performance has been computed as the difference between the performance prediction and the actual performance. In case the system predicted that a classifier achieves a higher performance than it actually obtained, this value is negative.

Figure 8.2 displays the absolute deviation means obtained by the different selection strategies, under different performance metrics. It can be observed that the voting strategies do not influence the final predictions very much. Weighted voting (W) obtains better results than democratic voting (D), but in most cases the difference is so small that it does not justify the additional resources needed to compute the weight of each neighbor. Moreover, the distance computation and neighbor selection strategies that obtain the smallest absolute deviations are, in order: Top 3 Chebyshev distance with Top 3 neighbor selection (C3-T3), Chebyshev distance with Top 3 neighbor selection (C-T3) and Euclidean distance with Above Middle neighbor selection (E-MID).

By analyzing the deviation mean results from figure 8.3, more details on the way each strategy combination works can be inferred. Top 3 Chebyshev distance with Top 3 neighbor selection (C3-T3) achieved negative deviation means for the accuracy and generalized geometric mean metrics. Thus, the strategy combination is overly optimistic on these metrics. The Chebyshev distance with Top 3 neighbor selection (C-T3) makes optimistic predictions on the exact same metrics. This strategy combination seems to be the best choice when

predicting classifier performance evaluated with the general metric, but is not appropriate for the other metrics in the system. The strategy combination with the best results on all metrics is Euclidean distance with Above Middle neighbor selection and democratic voting (E-MID-D). This combination obtains positive deviation means for most metrics, having a very small negative deviation for the geometric mean. This is the preferred behavior for the system.

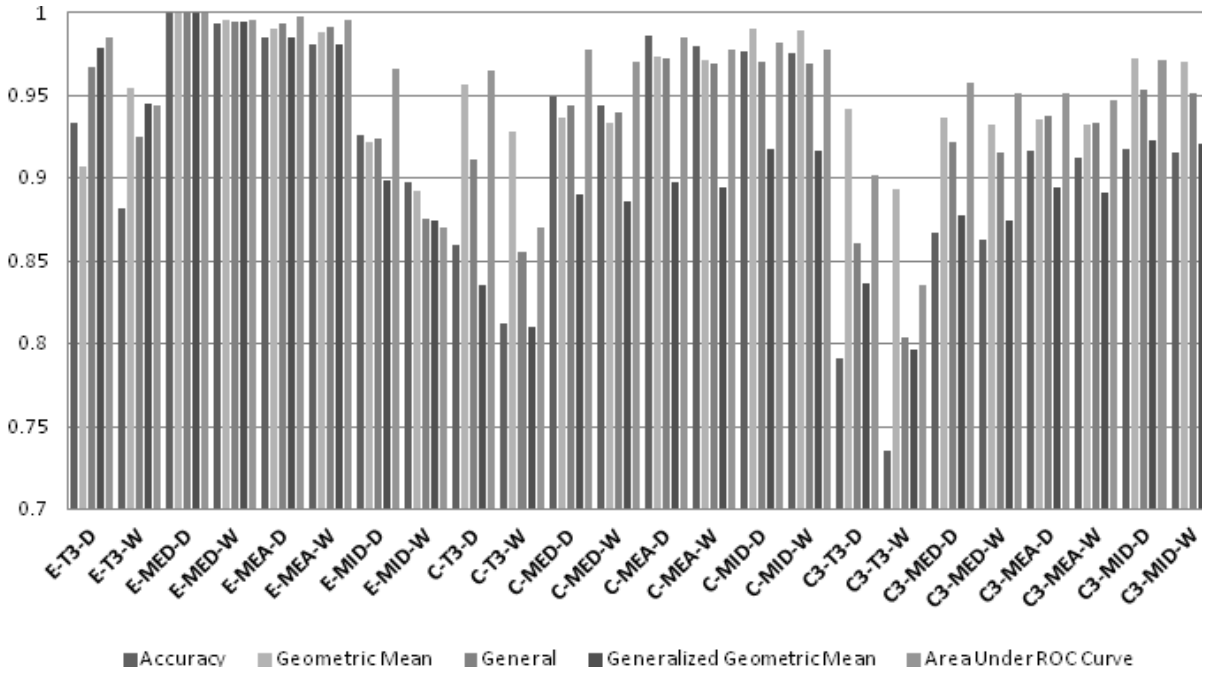


Figure 8.2 Absolute deviation means of different prediction strategies, under different metrics

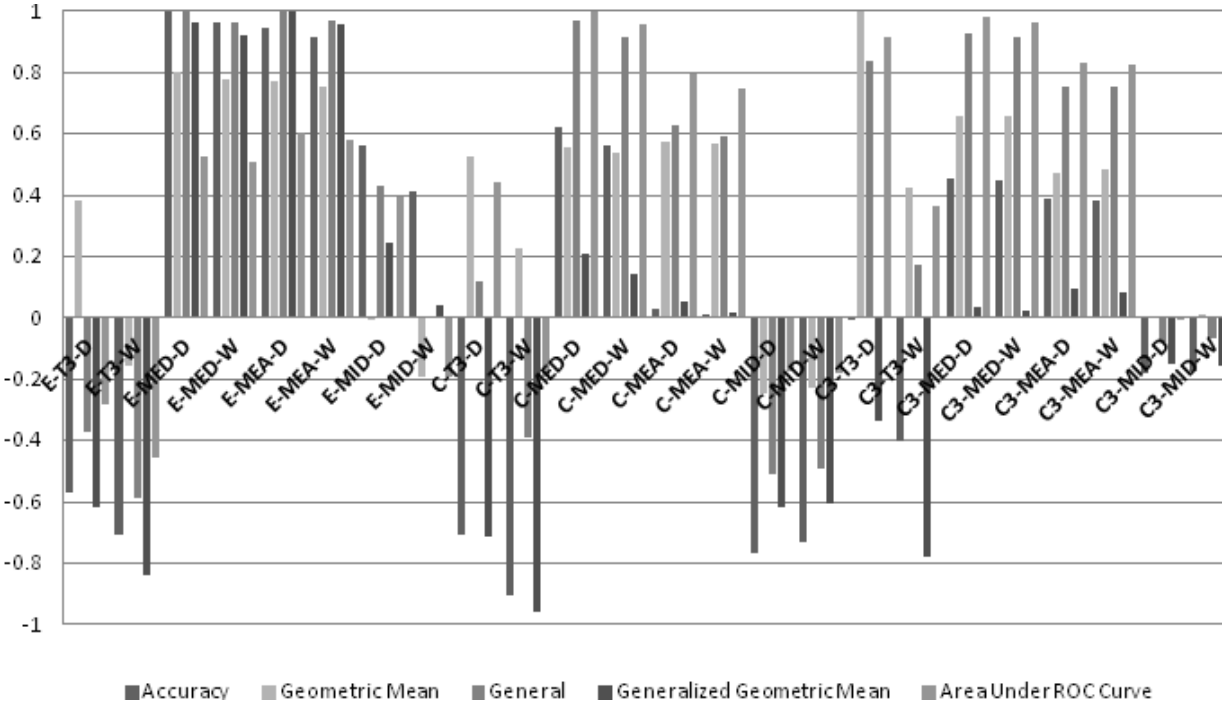


Figure 8.3 – Deviation means of different prediction strategies, under different metrics

In conclusion, the strategy combination with the best results on all metrics is Euclidean distance with Above Middle neighbor selection and Democratic Voting. It obtains positive deviation means for most metrics, having a very small negative deviation for the geometric mean. The tests also reveal that the voting strategies do not significantly influence the final results.

8.2 Enhancements by Data Partitioning

Multiple meta-learning methods which employ some kind of data partitioning strategy to evolve multiple sub-models, and combine the predictions to achieve increased classification performance and/or scalability, have been proposed in the scientific community [Bre96, Cha96, Fre97]. This section presents our work on the analysis of such an existing technique and several enhancements and proposes new strategies for data partitioning and combined prediction. Also, in the context of a specific DM application scenario, two case studies are presented, in which hierarchical classification models using data partitioning strategies are proposed, for two application scenarios: offline signature recognition and network intrusion detection.

8.2.1 The Arbiter-Combiner

In [Mer09], we have explored several strategies for improving the classification performance, via data partitioning. Based on the general arbiter method proposed in [Cha96], we have developed a new method, which considers several enhancements over the general scheme: automatic building of a perfectly balanced n-order tree; building a user specified structure of the tree; different learners on each node of a specified tree structure; different learners for arbiters and base classifiers with automatic tree building; data partitioning according to the tree structure; no union of subsets required. We have performed empirical evaluations on the new method, analyzing the impact of different parameters on the performance: leaf number, order of the tree (and implicitly the height of the tree and number of meta-classifiers), selection rule, and also of using a cost sensitive learning strategy instead of the traditional ones. Also, we tried to improve the accuracy for under-represented classes by applying random oversampling to the training data. We have also analyzed how the predictive performance of classifiers improves along each level of the tree. Training the classifiers was done using the same or different learning algorithms. The results indicated that arbiter trees have similar or higher predictive performance when compared to the individual classifiers, achieving a relative improvement of up to ~7% for the J4.8 classifier and of ~6.5 for the NB classifier. Binary arbiter trees obtained better results than higher order trees. The different-arbiter selection rule performs the best compared to the other selection rules and individual learners. Cost-sensitive learning on the leaves boosts the classification and/or balances the confusion matrix. Moreover, random oversampling balances the confusion matrix too. However, due to randomization, the predictive performance can sometimes degrade.

In the same work, an original classification method using data partitioning has been proposed: the arbiter-combiner, which specifies a regression model for each class, in each node, except for the leaf nodes [Mer09]. During the evaluations performed on the method, different order trees with different number of leaves were induced, in order to study the effect of the number of leaves on the predictive performance of the model. During preliminary testing, it was observed that too few instances propagate up to the meta-learner. The minimum number of meta-instances was restricted to the number of records for a base/leaf classifier. Two methods were tested to complete the training set for the meta-learner: include instances randomly and weight the meta-instances and add them in increasing order of their weight. The results have indicated that both binary and higher order trees yield good results. Also, the arbiter-combiner strategy is fitted for imbalanced datasets with multiple class labels.

In real-world classification problems (such as medical diagnosis) such a situation often occurs, and thus, the arbiter-combiner strategy might provide a more robust solution in these cases than the strategies which first binarize the problem and then apply customized methods for two-class imbalanced problems (as the ones presented in Chapter 7 of the current thesis).

8.2.2 A Hierarchical Model for Offline Signature Recognition

The handwritten signature is perhaps the most employed authentication mechanism in formal agreements, financial systems, various documents and artifacts. Despite several known limitations, such as the relatively reduced robustness to forgery or signature style variation when compared to more evolved biometrics authentication strategies, signature recognition systems are widely employed, especially in authenticating banking documents, due to their satisfactory performance per cost ratio. Just like any other DM system, a signature recognition system operates in two phases: training and recognition. In offline systems, the recognition phase does not occur in real-time, i.e. such systems do not employ data on speed of writing, acceleration or number of strokes in writing for each sample. However, their reduced cost makes them suitable for interpreting handwritten postal addresses on envelopes, sorting mail by postal code, automated reading of checks and tax returns or reading courtesy amounts on bank checks. The image processing and feature extraction steps are of vital importance in such systems, since they produce the data for the recognition process.

In [Bar09] an offline signature recognition system, which considered a wide collection of predictive features, employed separately by different other systems, has been proposed. Also, a new taxonomy of the types of features employed by such systems has been established and two new distance-based features have been introduced. Several iterations on the data mining process have been performed, completing the specific steps that were required by the specific problem requirements. Two feature selection strategies have been explored (a wrapper and a filter method), together with two classification strategies: Naïve Bayes and the Multilayer Perceptron. The Naïve Bayes classifier has been found to obtain the best results under the initial classification conditions, when compared to the Multilayer Perceptron – an accuracy of ~ 85%, compared to ~80%. Feature selection has further improved the recognition performance to ~91.4%, for the Naïve Bayes classifier. The analysis on the learning curves of the two classifiers indicated that the signature collection process should continue, since neither curve seemed to stabilize, when using less than 20 instances per class. With the addition of new data, increasing both the number of signatures/individual (i.e. instances/class) and the number of individuals (i.e. classes), the model based on a single multi-class classification step proved to possess insufficient separation capabilities (under-fitted the problem).

Therefore, in [Bar10, Pot11a], the previously proposed system has been enhanced by introducing a new method for hierarchically partitioning the training data using clustering. The aim of the method was to improve the recognition rate and the scalability of the system, by splitting the training data into smaller datasets, via clustering, and building classification sub-models for each split. When a new signature instance is to be classified, the hierarchical classifier first clusters it to find the best classification sub-model. It then presents the instance to the specific classification sub-model, which assigns the appropriate class label to the instance. Experimental evaluations have indicated that, for 76 classes and up to 20 instances/class, a number of 7-8 clusters generally yields good results. However, the optimal value for the number of clusters depends on the number of instances used for training. Also, several classifiers have been investigated for the classification step. The Naïve Bayes learner was selected as being the most appropriate in the majority of cases, providing accurate sub-models. Feature selection was again found to further improve the recognition performance,

achieving an increase of 1.62% on the accuracy against the initial model on a training set having 14 instances/class and 3.23% on a training set with 20 instances/class.

8.2.3 A Hybrid Approach for Network Intrusion Detection

A Network Intrusion Detection Systems (NIDS) represents a combination of hardware and software resources that are employed together in order to maintain system security by examining network traffic and detecting signs of potential violations of security policies in a computer network. NIDS's are divided into two categories: misuse (also known as signature-based) and anomaly detection systems. The first category encompasses systems which match traffic data against a database of known attack signatures, while anomaly detection systems employ learning models to separate between normal and anomalous traffic. Misuse systems are heavily based on human expert knowledge about real world attack patterns. Anomaly detection systems have an advantage over misuse systems because they can detect zero-day attacks or new attacks that are not part of an attack signature database. Also, misuse detection can impose noticeable overhead on systems when the network behavior matches multiple signatures.

In [Lem12c] a hierarchical model for network intrusion detection has been proposed, which combines the predictions of several binary classifiers at the first level and employs an additional classifier on the second level, specialized on classifying "difficult" instances – such as new types of attacks. The basic flow of the method starts with an initial data preparation step, in which several pre-processing operations are performed on the available training data, to improve its quality and generate the appropriate training sub-sets for the binary classification stage. Each binary classification module is specialized on correctly classifying a specific class of interest, and possesses the highest performance for that specific class. The predictions obtained from the binary classifiers are processed using a voting model which combines the individual predictions and generates the output prediction. In addition to existing voting strategies, a new method for prediction combination is proposed, which takes into account the data and error cost imbalance ((figure 8.4). More precisely, the predictions of the binary modules are initially ranked according to domain knowledge – data distribution and the gravity of failing to identify a specific class. Then, the instance to be classified is presented, in turn, to each binary model, until one of them produces a positive identification, i.e. the probability that the instance belongs to the given class is larger than the identification threshold value. The identification threshold values are learned for each individual binary model. Instances which cannot be classified via this mechanism (i.e. the voting strategy cannot indicate a certain output label) are delivered to the level 2 classification module. The conceptual architecture of the system is presented in figure 8.5.

The experiments were conducted on a dataset derived from the NSL-KDD Dataset [Tav09], which is an improved version of the KDD CUP '99 Dataset, having the following classes: normal and anomalous traffic. Redundant or incorrect data, as well as duplicate records have been removed from the initial dataset. The KDD CUP '99 Dataset has been pre-processed in a similar manner, to obtain a five-class dataset: the 4 classes of attacks corresponding to four main attack categories (DoS - Denial of Service, Probe, R2L - Remote to Local and U2R - User to Root) and the Normal class for non-attacks. Thus, starting from the initial training set, which contains 23 classes (22 types of attacks and normal traffic records), and testing dataset, with an addition of 17 new types of attacks, the attacks have been grouped into the above mentioned 4 attack categories. This resulted in a KDD+4 Training and Testing sub-sets, each containing 41 attributes, divided into 3 categories [Far09]. The first is represented by the duration, source bytes, destination bytes, service and TCP flags; the second category comprises of the features related to packet content and the last contains connection-related features, such as the connection time.

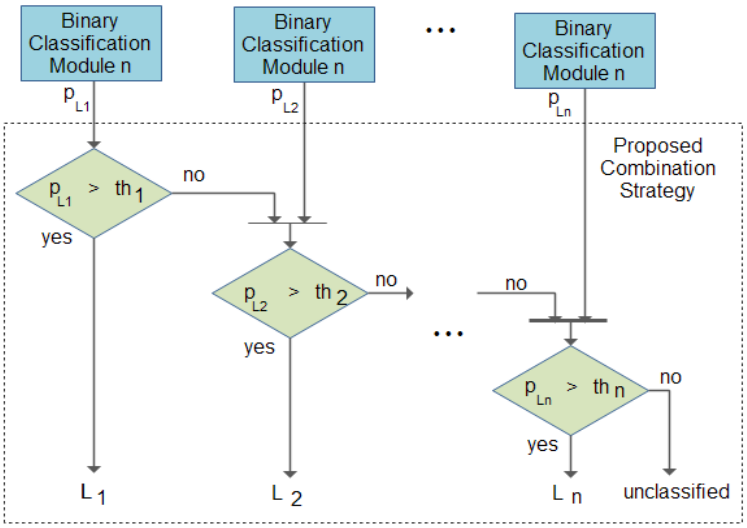


Figure 8.4 – Hierarchical prediction combination strategy

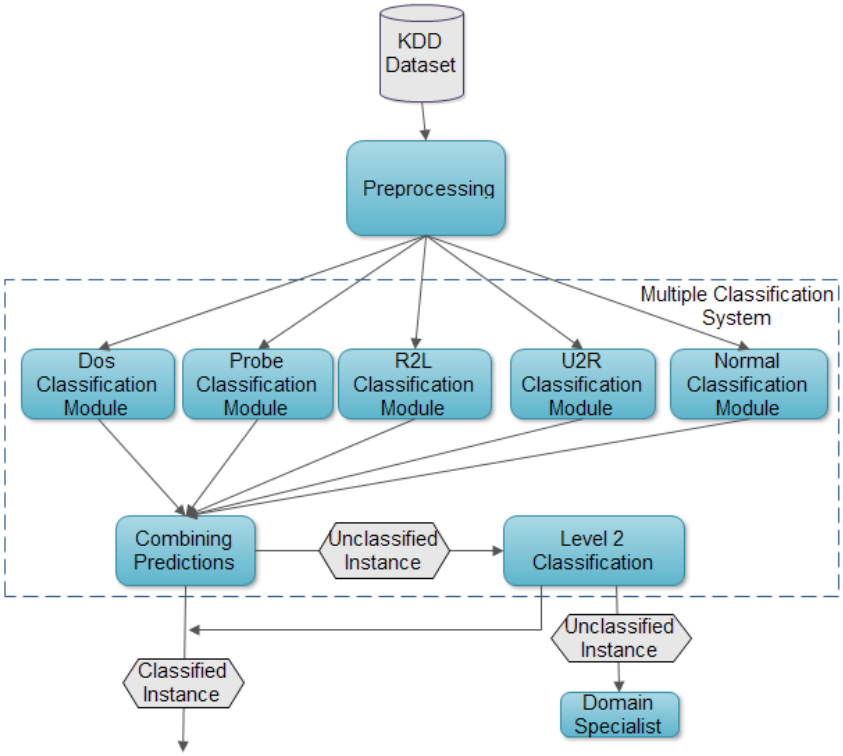


Figure 8.5 – Classification architecture

The resulting training and testing sets have been employed throughout the experiments and are further referred to as KDD+4 sets.

The major challenge with the KDD+4 training and testing datasets is that they have imbalanced class distributions. The class imbalance problem has been presented in Chapter 7 of the current thesis. Table 8.1 and Table 8.2 display the number of instances of each class in the training and testing datasets. The second row of each table shows the value of the imbalance ratio (IR) for each category of attack, relative to the normal traffic instances. It can be seen that the R2L class has an IR of 1 attack instance to 67.68 instances of normal traffic, but the highest IR appears for U2R, which has 1 attack instance to 1295 normal instances. When handling such a dataset, classification algorithms are biased towards the majority class due to its over-prevalence [Cha03].

Table 8.1 - KDD+4 Training Dataset

	DoS	*Probe*	*R2L*	*U2R*	*Normal*	*Total*
No. Inst	45927	11656	995	52	67343	125973
IR	1.47	5.78	67.68	1295		

Table 8.2 - KDD+4 Testing Dataset

	DoS	*Probe*	*R2L*	*U2R*	*Normal*	*Total*
No. Inst	7458	2421	2852	67	9746	22544
IR	1.31	4.03	3.42	145.46		

In addition, according to [Ngu08] and preliminary evaluations performed on the training set, no single classifier can achieve a high detection rate for all types of attacks. Thus, the focus is to obtain a high detection rate for each attack class, with major improvements for the two least represented attack classes R2L and U2R – for which similar systems in the literature achieved low recognition rates.

The tuning flow for the multiple classifier system on the first level is presented in figure 8.6. Each binary classification sub-model has to be built such as to distinguish, as best as possible, between a certain type of attack and normal packets. We have experimented with several classifiers and settings for each individual binary problem. Following a series of comparative evaluations, specific choices regarding the classifiers and settings, as well as sizing and finding the optimal distribution for each binary problem have been made. They are presented in the following. To determine the classifier that has the highest detection rate for a certain class, the 5 binary datasets have been generated: the first four containing all instances of a given attack type (as the positive class) and a sub-sample of normal instances (as the negative class), respectively, and the fifth containing all the instances of the normal (positive) class and a sub-sample taken from all attack types (negative class). The tests have been performed using 5-fold cross validation and default classifier parameter settings, on nine different classifiers belonging to different categories, employed previously by similar systems. The top two classifiers which achieved highest TP_{rates} and lowest FP_{rates} for each sub-problem are considered for further testing. Thus, the following classifiers have been selected for subsequent evaluations: REPTree and RandomForest for the DoS module, RandomForest and NBTree for Probe, BayesNet and NaïveBayes for R2L, NaiveBayes and BayesNet for U2R, RandomForest and NBTree for Normal.

Modifying the distribution and volume of the dataset is done by using re-sampling techniques. This set of tests has been performed by varying the distribution of the primary class from 10% to 90%, with a 10% increment. Two different re-sampling strategies have been explored: simple re-sampling and smart re-sampling. Simple re-sampling performs random under-sampling on the majority class while oversampling randomly the minority class. Smart re-sampling, on the other hand, performs oversampling by artificially generating minority class instances using the SMOTE algorithm [Cha02]. The tests have been run on the previously selected classifiers configured with the default parameters. The classifiers were trained on 40% of each binary training set and validated on the remaining 60%. In order to compare the results the F_β-measure was employed, with $\beta=2$ for the strongly represented classes (DoS, Probe, Normal) and $\beta=4$ for the weakly represented classes (R2L and U2R).

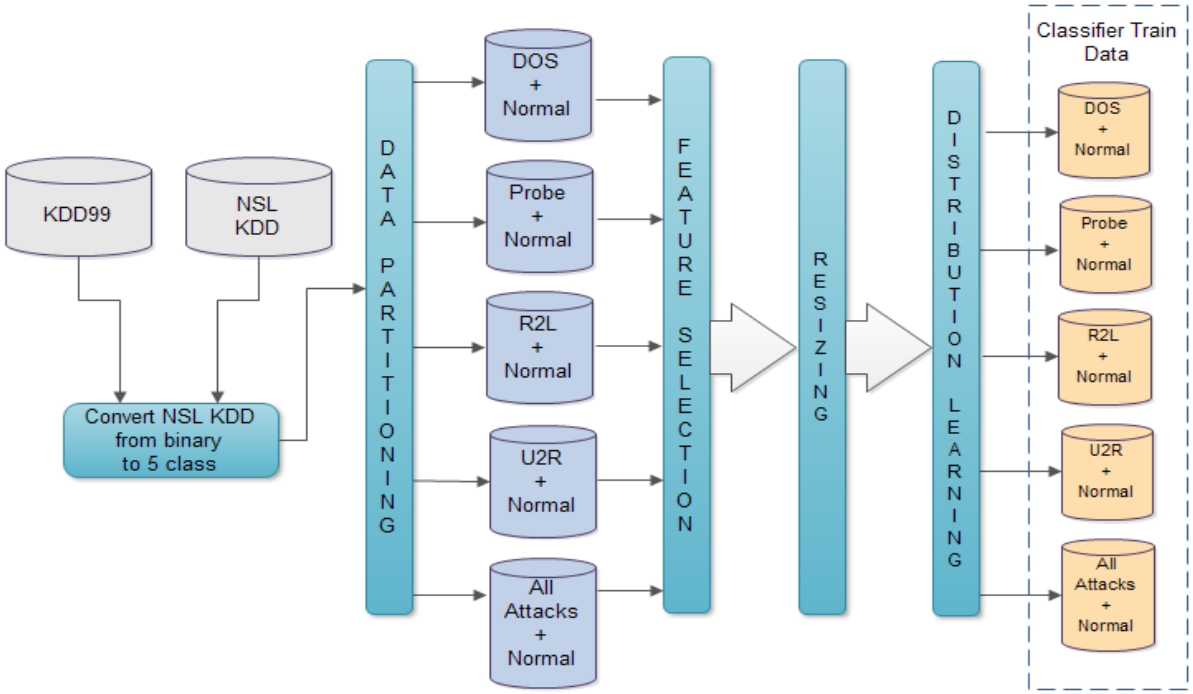


Figure 8.6 – Tuning stages for the multiple classifier system

We have selected this metric to assess the results since it is easier to compare between different performance levels than analyzing the confusion matrix; moreover, by varying β different importance levels can be assigned to the different attack classes. The results indicated that the best distribution depends heavily on the problem at hand and the employed learning method (table 8.3). Only in some of the cases the smart re-sampling produces superior performance to the simple re-sampling strategies. If we take into account also the speed of each approach, the employment of smart re-sampling techniques over simple strategies is not justified.

For tuning the parameters of the classifiers employed in the binary modules, 5-fold cross validation was employed, on the naturally occurring distribution. The best choices for each module are: 16 trees and 11 attributes for the RandomForest in the *Probe+Normal* module; for the *DoS+Normal* module, 4 trees and 22 features for the RandomForest classifier; for *R2L+Normal* and the *U2R+Normal* module the default BayesNet settings; for the *Normal+Attack* module, 20 trees and 14 features.

Table 8.3 – The best learning distributions for each module

		Simple resampling		**Smart resampling**	
		% minority	*F_β-measure*	*% minority*	*F_β-measure*
DoS	REPTree	50	0.9997	40	0.9996
	Random Forest	80	0.9999	70	0.9999
Probe	Random Forest	30	0.9977	40	0.9971
	NBTree	50	0.997	20	0.9962
R2L	BayesNet	10	0.9578	10	0.9465
	Naïve Bayes	30	0.7264	30	0.7204
U2R	Naïve Bayes	90	0.2382	60	0.3597
	BayesNet	20	0.6607	20	0.6492
Normal	Random Forest	70	0.9988	80	0.9989
	NBTree	70	0.9986	80	0.9983

Table 8.4 - TP and FN values obtained by the different voting strategies

	DoS		*Probe*		*R2L*		*U2R*		*Normal*	
	TP	FN	TP	FN	TP	FN	TP	FN	TP	FN
Maj	22587	4969	3814	3179	41	556	1	30	40401	4
Avg	22709	4847	3793	3200	43	554	1	30	**40402**	3
Max	27535	21	6761	232	462	135	5	26	40393	12
Med	6	0	22	0	25	0	11	0	0	40354
Prod	26869	20	3720	216	4	134	0	26	40393	11
Hierarchical	**27556**	0	**6981**	12	**593**	4	**28**	3	39954	451

Several different voting strategies have been considered for combining the individual predictions of the binary classification sub-modules: majority voting (Maj), product voting (Prod), average voting (Avg), maximum voting (Max), median voting (Med) – all available in WEKA and the proposed voting strategy (Hierarchical).

All the configurations previously identified have been employed to train the overall system. The results obtained by evaluating the fully configured system on the test set can be seen in the first column of the Table 8.5. The results obtained by our system have been compared to other systems evaluated on the KDD'99 datasets. The proposed hybrid system yields significant improvements in the detection of minority classes compared to the other systems [Gog10, Elk00]: 90% correctly labeled instances for the R2L class and 85% for the U2R. These results are achieved with the level 1 multiple classification method alone; the level 2 classifier is currently under development. We expect its addition to produce further improvement in the recognition of attack instances.

8.3 Speedup and Scalability through Parallel/Distributed Strategies

There are situations in which specific problem requirements impose additional constraints on the DM process. For example, when dealing with extremely dynamic distributions (e.g. spam sdetection, network intrusion detection, etc) periodic re-training and model fitting are necessary, making training time also a potential constraint. By exploiting the parallelization potential of existing techniques, faster methods can be developed to respond to such demands. In this sub-section we present our efforts to achieve speed-up through parallelization on two important DM techniques, widely employed in the present thesis: decision trees and genetic search. A parallel version of the Evolutionary Cost-Sensitive Balancing method proposed in Chapter 7 is also presented.

8.3.1 dECSB – Distributed Evolutionary Cost-sensitive Balancing

In [Lem12a], an original distributed version of the ECSB method proposed in Chapter 7 has been developed. The necessity for such an approach has been dictated by the inability to produce learning models, via ECSB, for relatively slower algorithms, such as SVM, on slightly larger datasets (~5000 instances). This is because fitness evaluation within the GA implies training and evaluating a cost-sensitive meta-classifier in a cross-validation loop.

Table 8.5 - Recognition rates/classes

Class	Our System	KDD Winner	Catsub	FCM	SVM +DGSOT
DoS	97%	97%	100%	99%	97%
Probe	100%	83%	37%	93%	91%
R2L	90%	8%	82%	83%	43%
U2R	85%	13%	0%	0%	23%
Normal	89%	99%	82%	96%	95%

Therefore, the complexity of the method depends on: the population size (*N*), the number of generations (*I*) and the complexity of the cost-sensitive classifier, which, in turn, depends on the complexity of its base classifier; since the cost-sensitive classifier intervenes only at prediction time, by altering the predictions according to the given cost matrix, its time complexity is dominated by the complexity of the base classifier employed; thus, the leading term for the time complexity of ECSB is:

$$T_{ECSB}(m, n, N, I, k) = N \times I \times (k \times T_{base}(m, n)) \quad (8.2)$$

where *k* is the number of folds in the cross-validation loop, *m* is the number of instances, *n* is the number of attributes in the training set. T_{base} can be further divided into training and evaluation time complexity – most algorithms are computationally-intensive for the training stage; an exception to this is the kNN classifier, or other "lazy" methods (i.e. which don't build a prediction model).

The distributed algorithm has five key elements: *a candidate factory*, *evolutionary operators* (crossover and mutation), *a fitness evaluator*, *a selection strategy* and *a termination condition* (as depicted in figure 8.6). The candidate factory is responsible for generating the population of *n* candidate individuals. The genotype of an individual consists of two chromosomes associated with the misclassification costs and two chromosomes representing parameters of the base classifier used; these chromosomes are encoded in Gray; their phenotypes are expressed as integers.

The evolutionary operators – crossover and mutation – are applied at bit level for each chromosome. Multiple point crossover and single bit uniform mutation have been considered, in the attempt to increase search variability. As different problems may require separate evaluation strategies, we have assessed several evaluation metrics as *fitness functions*, both balanced and imbalanced. In Chapter 7, we have concluded that the geometric mean, the balanced accuracy and the linear combination between the true positive rate and precision are the most appropriate under various settings, for different domains/problems. Fitness evaluation is the major point for parallelization in our approach.

For the *selection strategy* used to filter the individuals in the current generation for the evolution of the future generation, an elitism strategy has been considered. The amount of individuals from the old generation which will be kept in the new one can be specified. This mechanism has been employed in conjunction with the increased search variability, to improve the robustness of the search, by encouraging the exploration of the search space rather than favoring early convergence to highly fit individuals. Two *parent selection* methods have been employed: rank selection and roulette wheel selection. Rank selection provides an effective mechanism for controlling the selective pressure.

The end of the evolution cycle is controlled through the *termination condition*, represented by reaching the predefined number of evolution cycles or a certain fitness level.

In the sequential ECSB, when evolving from one generation to another, fitness computation represents the main bottleneck. Two types of "data" are utilized within ECSB: (1) the instances in the training set used to build the classification model and (2) the individuals in the genetic population (cost matrix and base classifier settings). Accordingly, two distribution strategies have been considered.

The first approach is computation-driven (figure 8.8): the training set is replicated in a distributed environment capable of splitting the GA population in multiple subsets of individuals and performing parallel fitness evaluation for each subset of individuals (as depicted in Fig. 3).

The time complexity of this approach can be expressed as:

$$T_{dECSB}(m, n, N, I, k, r) = N * I * \frac{(k * T_{Base}(m, n))}{r} + T_{overhead} \quad (8.3)$$

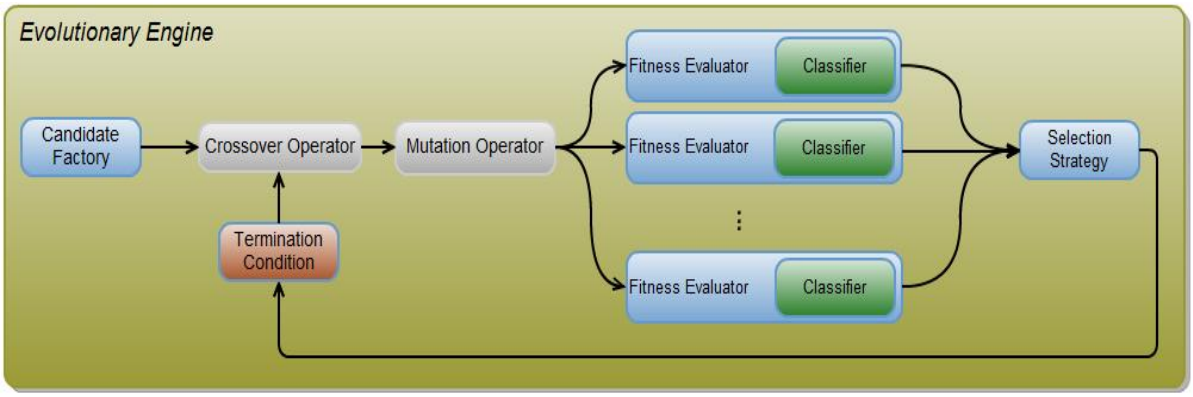


Figure 8.7 – Conceptual architecture of the Evolutionary Engine

where *r* represents the number of processing nodes, or, otherwise said, the splitting ratio, and $T_{overhead}$ represents the overhead introduced by splitting the problem and combining the results. Generally, the first term is significantly larger than the second. This method should provide the same quality of solutions as the sequential ECSB. As a possible limitation, since the complexity of model generation and evaluation for a particular classifier is strongly influenced by the size of the training set, when handling large training-sets this approach it might not scale.

The second approach, which aims to avoid this drawback, is data-driven (figure 8.9): the training set is divided into several partitions, keeping the same distribution in each partition. The entire population is evaluated in parallel for each partition, and the fitness value for an individual is computed as the average of the values evaluated on each partition. The time complexity of this second approach can be expressed as:

$$T_{dECSB}(m, n, N, I, k, r) = N * I * (k * r * T_{Base}(\frac{m}{r}, n)) + T'_{overhead} \quad (8.4)$$

where *r* is the number of computation nodes and $T'_{overhead}$ is the overhead corresponding to the distribution via this approach. Again, the leading term is the first one. It is easy to prove that the data-driven approach has a lower time complexity than the computational-driven one and thus we expect it to be faster and scale better to large datasets. The question with this strategy is whether the overall performance of the solution is affected, due to the lack of sufficient representative instances in some partitions. Which of the two approaches is better depends on the specific problem characteristics.

The development context for our proposed solution plays an important part in the success of the deployment. The major prerequisites are: a background capable of sustaining distributed computations on large training sets. Such an infrastructure should expose its features to a more abstract layer were the conception of data and computation intensive solutions are not dependent on low level infrastructure issues.

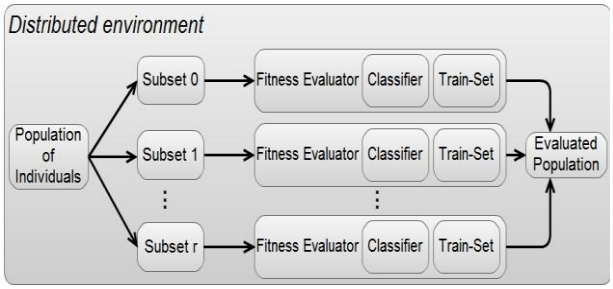


Figure 8.8 – Computation-driven approach

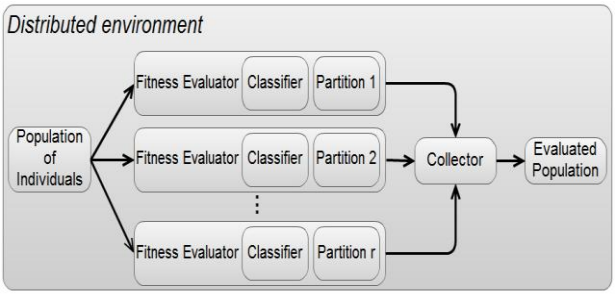


Figure 8.9 – Data-driven approach

On top of these two layers, a framework for developing evolutionary algorithms is required. In figure 8.10, the layers discussed above are illustrated, with reference to the existing frameworks employed to sustain our approach.

For the outmost layer Hadoop[3] has been selected. It provides a distributed file system and also a distributed computation engine based on the map/reduce paradigm, for performing real time computations in a scalable context. From a practical point of view, we use Hadoop map tasks for evaluating an individual or a population of individuals. This is done by writing Java-based Hadoop map tasks where WEKA is used for performing the actual evaluation. The basic flow covers: deploying the map task to a node where the dataset is present, loading the dataset, creating a WEKA evaluator, training the evaluator on the loaded dataset, evaluating the created model(s) and expose the evaluation result(s) in the Hadoop context. The reduce task implementation depends on the distributed fitness evaluation strategy used.

On the second layer we need a solution able to distribute our fitness evaluation. The Mahout[4] framework provides a possible solution for this layer, making all configuration and implementation details transparent to the developer when writing a scalable machine learning application. One feature of Mahout is the possibility of running distributed genetic algorithms implemented in the Watchmaker[5] framework. For the second approach in distributing fitness evaluation (data-driven fitness evaluation) we developed our own solution. It is a Java-based implementation of Hadoop and Watchmaker interfaces, acting a bridge between Hadoop and Watchmaker.

For the third layer, the Watchmaker framework is used, which supports the development of scalable GA's, by providing a non-invasive API and a reasonable collection of specific GA mechanisms. From a structural point of view, our solution is mapped onto the Watchmaker architecture, using some components already available within the framework.

We are currently performing a series of evaluations, comparing dECSB to the sequential version of the method and to other techniques for imbalanced classification. Preliminary results indicate that dECSB improves the performance of the classifiers, under almost all the metrics considered. Also, tuning the GA parameters is likely to produce further improvements in performance. Time measurements indicate that the data-driven approach is the most appropriate of the two strategies adopted [Lem12a].

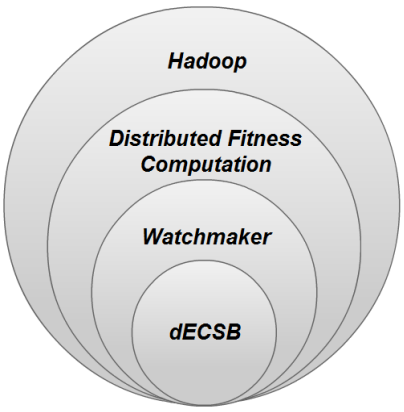


Figure 8.10 – Architectural context

[3] http://hadoop.apache.org/
[4] http://lucene.apache.org/mahout/
[5] http://watchmaker.uncommons.org/

8.3.2 A Parallel Decision Tree

A serious limitation of decision trees, rooted in the splitting mechanism, is the memory requirement. In order to compute the value of the splitting function for an attribute, all training instances need to be processed. As the size of the training set increases, loading all instances into the main memory at once becomes impossible. Several solutions address this shortcoming, the most prominent being the SPRINT („Serial PaRallizable INduction of decision Trees") algorithm [Agr96] and SLIQ [Meh96]. Starting from the SPRINT decision tree, an original parallel version of the algorithm, with several proposed improvements over the sequential version has been proposed [Dod11]. The splitting criterion employed is the Gini index and pre-pruning was used as control mechanism for the size of the resulting tree. The method eliminates the memory issue by using the auxiliary structure proposed in SPRINT – the attribute list – and maintaining the data in a relational database. Speed-up is achieved by evaluating each possible split point in parallel.

The experiments performed have shown that the parallel implementation outperforms the sequential implementation on each platform considered, achieving significant speed-up factors (up to 3 times faster on a quad core). Moreover, the results have indicated that by increasing the number of threads which can be generated at a time, the performance of the parallel implementation improves significantly (the parallel implementation achieved 4x speed-up on a quad core compared to the dual core).

8.3.3 A Lightweight Parallel Genetic Algorithms Framework

In [Fei11] a series of genetic mechanisms, which have later been employed in a lightweight parallel genetic framework, have been introduced. In the above cited work, the developed mechanisms have been presented in the context of providing efficient solutions to NP-complete problems. Therefore, a parallel genetic algorithm has been developed, using the *General-Purpose Computation on Graphics Hardware* (GPGPU) paradigm and the available *NVIDIA CUDA*[6] parallel computing platform and programming model. An island-based approach for the genetic search engine has been investigated, to isolate populations within a block and minimize inter-block communications. This is desirable for both the algorithm and the GPU-based implementation: at algorithm-level, subsets of individuals search for an optimum, while at implementation-level no inter-block synchronization is required, each island evolving separately in a block. Each population is mapped on a CUDA block, and each genotype in the population on a thread. The entire algorithm is run on the GPU to minimize the communication between the GPU and CPU. The result is passed to the CPU at the end of the execution. During the evolution, each thread generates an offspring from two selected parents, using crossover and mutation operators. Parent selection is roulette wheel with scaling on the fitness scores.

Evaluations on several traditional NP-complete problems existing in literature have been performed. The results have indicated that the parallel GPU-based GA implementation achieves significant speed-ups when compared to a sequential GA implementation, while maintaining the solution quality.

8.4 Domain-specific Data Mining Process Instantiations

8.4.1 User Type Identification: Static and Dynamic User Profile in Adaptive E-learning Systems

Adaptive e-learning systems represent a new paradigm in modern learning approaches. In the attempt to preserve the quality of the teaching effort (as in face to face

[6] http://www.nvidia.com/object/cuda_home_new.html

education), current trends in this area focus on improving the user's performance throughout the learning process, by stimulating faster learning and helping students develop new, desirable learning abilities. This is achieved by continuously monitoring and adapting the process to the user's capabilities, needs and desirable behavior. In [Fir09] we have proposed and designed an adaptive e-learning system which ensures content adjustment according to the user profile, similarly to face to face education. The system performs user profiling and employs concept maps to adapt the content accordingly. The design specifies both static and dynamic features for characterizing the student's profile. The strength of the adaptive e-learning model is based on the intelligent component, which defines rules based on both matching and mismatching between teaching strategies and learning styles. It is continuously evaluating the user profile, and adapts the content to improve the student's performance. Four different types of users were considered, according to the most prominent findings in literature [Ver96]: understanding based, memorization-based, concrete and non-directed.

The intelligent component for the adaptive e-learning system has been developed by performing several iterations of the data mining process. Static data has been collected from a sample of 304 Romanian students in the engineering field, using a psychological test. The answers are partitioned, according to psycho-pedagogical knowledge, to form intermediate attributes, which determine an initial classification of the user. A Bayes Network (BN) static model of the user was initially assessed. The BN model evaluated the percentage of membership of each individual to each of the four learning styles, yielding narrow separation boundaries between the four classes. In the attempt to achieve a better separation of the typologies, several experiments have been conducted with the k-means clustering technique, using three different values for k: 3, 4 and 5 [Tri10]. The results suggested that a 3-type based analysis explained the data better than the theoretical 4-type based. The few discrepancies observed have been attributed to two sources: (1) the cultural and regional differences between the populations evaluated and the ones reported in literature and (2) the non-uniform distribution of the population evaluated (all the individuals were from the same study year).

In [Lem11a], the quality of the static user model has been assessed through various unsupervised techniques. The sub-goals were to compare and contrast different models – a clustering technique with a Self-Organizing Map (SOM) approach and to analyze the feature – user type correlations. Again, a 3-type model has been found to best fit the static user profile. Also, a series of attribute-user type correlations have been identified in the data sample. A design which integrates the static and the dynamic user profiles into a common approach has been proposed (figure 8.11). It consists of two sub-components: a k-means clustering component and a SOM. The first it is used for initial user type identification based on static attributes. Its results are used for initial SOM training and evaluation. The input of the SOM component has two parts: a static part (the static attributes, as recorded by the initial psychological test) and a dynamic part (the attributes identified during the user's interaction with the system).

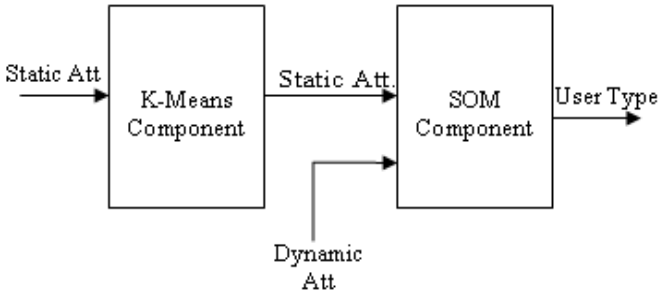


Figure 8.11 – User type identification component

The values of the dynamic attributes may change over time, as the user learning style evolves; however, the static attributes are not re-assessed (the pre-test is presented to the user only once, during his/her first interaction with the system). The SOM component training process is continuous – the map configuration may adjust as the users interact with the system, and the values of their dynamic attributes change.

The SOM component is initially built using only the static attributes; all the dynamic attributes have the value zero. After this initial construction phase, the learning rate and the neighborhood radius of the SOM are set to low values in order to ensure that changes due to dynamic attributes are made in time, representing a consistent user behavior. This is necessary because the user learning style cannot change abruptly and, therefore, a spike in the user's dynamic behavior does not produce major changes to the map topology. However, if the same modified behavior persists in time, this means that the user learning style has evolved and this will have an effect on the map topology and user classification. This desirable behavior is ensured through another mechanism: user history. The values of the dynamic attributes represent aggregates of the actual recorded values at different moments. More recent interactions weigh more, and the influence of distant interactions is minimal, or absent. Otherwise said, a decay function will be employed to aggregate the values of different user interactions with the system. In time, due to an increasing number of users being clustered based on their dynamic attributes especially – since the static attributes don't change their values, the SOM component evolves and the influence of the dynamic attributes in the user type identification increases.

Thus, the employment of SOMs allows for continuous updating of the user model, while k-means clustering provides an initial assessment. An evaluation on artificially generated data has been performed, to study how the two components (static and dynamic) are integrated in the proposed model. The results have indicated a smooth passage between the static and the dynamic components – the SOM built on the static attributes is slowly modified by the introduction of the dynamic attributes (i.e. the user starts to interact with the system), but the separation between the learning styles remains well defined.

8.4.2 Handwriting Recognition: Historical Documents Transcription using Hierarchical Classification and Dictionary Matching

Historical documents are difficult to read and require the expertise of trained specialists, mainly due to the vast number of sources, quality of the target manuscripts and numerous existent scripts [Min01]. In practice, transcriptions are performed by hand, by a scholar specialized in the target form of writing (paleographer) and take a considerable amount of time and effort; hence the need for a flexible system capable of performing a (semi-)automatic transcription of such documents, while taking into problem-specific constraints.

In [Lem12b], a solution which aims to translate documents written in a difficult alphabet (the German script *Kurrent Schrift*) has been proposed. It does not aim to restore severely damaged manuscripts. It assumes that the text is visibly separable from the background – though most historical documents are expected to present some imperfections (paper folding, material aging).

We have identified three major stages – Document Processing, Word Processing and Word Selection, connected in a linear pipeline (as seen in Figure 12). First, the document image (i.e. the scanned version of the historical handwritten document) is processed by the ***Document Processor***. After separating the text from the background through a two-step procedure and removing malignant noisy areas, the document is de-skewed to improve the correctness of subsequent processing steps. It is then successively partitioned into lines of text and individual words.

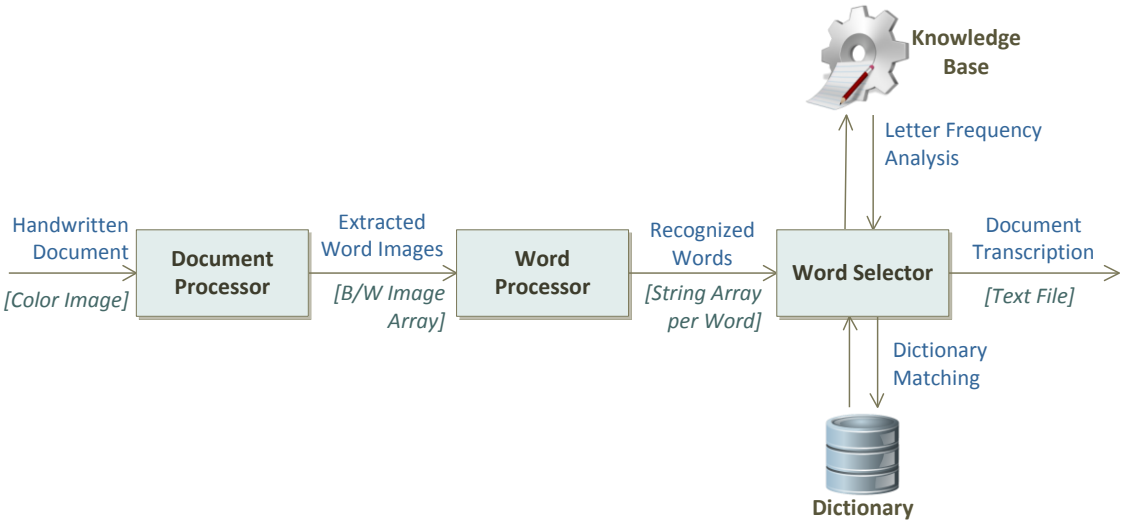


Figure 8.12 – System conceptual architecture

The ***Word Processor***, the core component of the system, finalizes the preprocessing of the input word images, by performing slant correction and character splitting. The shape of the binary character objects is then captured using a skeletonization filter, and important features that discriminate the characters are extracted. A classifier identifies each character and word variants are constructed.

The words are validated by the ***Word Selector*** using a local dictionary database and a Knowledge Base, generating transcription variants with attached probability. Inappropriate matches are pruned and the words reordered such as to generate the final transcription, the output of the system.

Preliminary experiments with the letter recognition module, within the word processor, indicate that the performance of a single-model recognition system is insufficient, the accuracy being too small for such an automated translator. Further experiments run over the best selected classifiers and analysis of the extracted results revealed that predictable confusions are produced amongst certain groups of characters. The groups of confusable characters are constructed based on pair-wise confusions between any two characters belonging to the same group. For this reason, a character can either be directly identified by the classifier, or belong to one or more confusable groups. Therefore, the groups do not form a partition of the alphabet, nor are they disjoint. Thus, a composite two-layer model is more appropriate for the letter identification step.

A multi-class single classifier is employed at the first level, responsible for discriminating amongst non-confusable characters, as well as propagating instances predicted as belonging to confusable sets to the second layer. Level two model comprises of an array of individually constructed classifiers, each responsible for discriminating only inside a distinct group of confusable letters.

Significant numerical features are extracted from the character image shape resulted after a series of image processing operations have been applied to the document scans [Lem12b]. In order to better discriminate among the characters, the following mix of features is considered [Vam09]: projections, profiles, transitions and zone densities. Because histogram-based features are dependent on the character image resolution (width, height), we propose histogram-compression based on the Discrete Cosine Transform. This approach captures the shape of the histogram. The resulting sequence is then normalized in the [0,1] interval, and only the first few coefficients are considered. This ensures both a fixed-dimension feature vector and noise reduction (generally located in the higher part of the signal's spectrum).

Parameter tuning is possible by varying the number of considered harmonics (coefficients), the optimal truncation index being determined empirically.

Several experiments have been performed to identify the best level 1 classifier for the word processor components. Moreover, parameter tuning has been considered in order to obtain more fitted models, for the selected classifier. The extracted letter dataset is a balanced dataset (25 classes with approximately 50 instances per class) having 37 features (represented by the computation of 4 Discrete Cosine Transform (DCT) components). Stratified 10 fold cross-validation has been employed for these tests, with accuracy as performance metric, since all classes are of the same interest and equally represented. The best results have been obtained by the SVM (82.12%) and the MLP(75.28%) classifiers. J48 (53.18%) and Naïve-Bayes (66.08%) performed well below the expected accuracy, the latter giving even worse results after a boosting mechanism was applied. The components of the system presented in this section are currently in different stages of development and evaluation.

8.4.3 Social Mining: Distributed Community Detection in Social Networks Using Genetic Algorithms

One of the current important research topics in data mining applications, triggered by the emergence of the social network phenomenon, is community detection. One interesting sub-problem is the identification of the community structure. In [Hal10] a distributed community detection approach has been developed, using genetic search mechanisms on top of Apache Hadoop. The algorithm uses the genetic representation proposed in [Par98]. An analysis of two possible fitness functions is provided, both from the point of view of efficiency and adaptability to a distributed context: the community score function proposed in [Piz98], and a new fitness function, which switches from the network modularity score to the community score measure during the evolution. This new fitness function represents an original contribution. The Apache Watchmaker framework has been employed for implementing the algorithm, on top of the Mahout Machine Learning library, which provided the possibility of running the genetic algorithm implemented in the Watchmaker framework in a distributed manner. The fitness evaluators were encapsulated using the infrastructure provided by Mahout and distributed on Hadoop nodes.

During the evaluations performed on well-known real-world networks, the community structure obtained by running the algorithm with the two fitness functions has been compared with results obtained by similar approaches, reported in literature. Several parameter settings for the genetic search algorithm have been investigated. The scalability of the approach has also been evaluated. The results have indicated that the proposed fitness measure achieves similar results to the community score-based fitness function, but reduces the running time significantly. Also, the distribution of the genetic algorithm has proved beneficial, since it has made possible to consider larger subsets of the input data and thus discover more complex relationships [Hal10].

8.4.4 Opinion Mining: Semi supervised opinion mining with lexical knowledge

Opinion prediction from text, i.e. automatically detecting the positive or negative attitude towards a topic of interest has received increasing attention over the last years. The proliferation of user-generated content on the internet has triggered the interest towards large scale mining of public opinion. To that end, we have examined an approach for building a graph-based semi-supervised sentiment polarity classifier, where a knowledge base for opinion mining is provided [Bal10]. In the development flow, we have created our own dictionary, using SentiWordNet, for tagging the words in the documents with the corresponding polarity. Then, we have implemented a semi-supervised classification algorithm, using the sentiment dictionary, similarly to the approach available in [Sin08]. To assess the flexibility of the resulting classifier, we have performed several evaluations on

review datasets from various domains. The results have indicated that unlabeled data can improve the classification performance in some cases. Time, processor and memory usage have also been considered, since opinion mining is a complex problem in essence, and the input data is usually large. The system can be easily integrated with the Rapid Miner tool.

8.4.5 Spam Detection: A Filter

In recent years, anti-spam filters have become ubiquitous tools in fighting this new but continuously growing phenomenon – spam. Whether we talk about internet service providers, or large organizations, e-mail providers or even individuals, everybody is interested in keeping the irrelevant, inappropriate and unsolicited emails to a minimum. Thus, we have developed a prototype for a new spam detection filter, which employs the kNN algorithm for the classification module, on a set of frequency–based features [Fir10]. Training set re-sampling was employed to determine the most appropriate learning distribution. A data update mechanism was considered to enable the system to employ the information coming in new emails, and thus continuously improve its performance. User feedback, used to correct misclassifications has also been implemented.

8.5 Conclusions

Specific domains may impose additional constraints on the general DM flow. Several non-functional requirements (such as model generation time, model complexity and interpretability) may also be of importance. Also, depending on the problem particularities, the general flow of the data mining process may be altered to accommodate for the specific domain needs. This chapter investigates several such scenarios, and attempts to provide viable solutions, at a level of generality such as not to be restricted to the specific domains they have been initially designed for.

The original contributions presented in this chapter include:

- The proposal and evaluation of a cascaded method for classifier baseline performance assessment, using the Dempster-Shafer theory of evidence
- An original meta-learning framework for automated classifier selection, which employs various meta-features and several different prediction strategies, while reducing user involvement at a minimum
- A composite metric for general classifier performance assessment
- A hierarchical model for classification problems having a large number of classes, based on clustering and classification sub-models, with application to offline signature recognition
- A hierarchical model for classification of multi-class imbalanced problems, which consists of a multiple classifier on a first level, and a classifier which focused on labeling difficult cases on the second level; the system has been applied to a network intrusion detection case study
- A distributed version for the original ECSB method proposed in Chapter 7
- The proposal and evaluation of an original classification meta-technique, based on data partitioning: the arbiter-combiner
- A model for static and dynamic user-type identification in adaptive e-learning systems
- A model for historical documents transcription based on hierarchical classification and dictionary matching
- An original system for distributed community detection, using genetic algorithms
- A dynamic composite fitness function for the community detection problem

- An original parallel version of the SPRINT decision tree classifier
- A lightweight GPGPU parallel genetic framework
- An original system for opinion mining, using semi-supervised learning with lexical knowledge
- An original spam detection filter, with learned training distribution

The original user profiling method proposed in section 8.4.1 is the result of research supported by PNII grant no.12080/2008: SEArCH – Adaptive E-Learning Systems using Concept Maps.

The original methods presented in this chapter have been validated through the publication of 3 journal papers:

1. Potolea, R., **Vidrighin B. C**., Trif, F., "Intelligent Component for adaptive E-learning Systems", *The Automation, Computers, Applied Mathematics Journal*, Vol. 18, pp. 270-275, 2009
2. Potolea, R., Trif, F., **Lemnaru, C**., "Enhancements on Adaptive E-learning Systems", *The Automation, Computers, Applied Mathematics Journal (ACAM)*, Vol. 19, no.3, pp. 475-482, 2010
3. Potolea, R., Trif, F., **Lemnaru, C**., "Adaptive E-Learning Systems with Concept Maps", *Revista Romana de Informatica si Automatica*, vol. 21, no.4/2011, pp. 43-56, 2011

2 book chapters:

1. Potolea, R., Barbantan, I., **Lemnaru, C**.,"A Hierarchical Approach for the Offline Handwritten Signature Recognition", *Lecture Notes in Business Information Processing* , 2011, Volume 73, Part 3, 264-279, DOI: 10.1007/978-3-642-19802-1
2. Potolea, R., Cacoveanu, S., **Lemnaru, C.,** "Meta-learning Framework for Prediction Strategy Evaluation", *Lecture Notes in Business Information Processing* , 2011, Volume 73, Part 3, 280-295, DOI: 10.1007/978-3-642-19802-1

and several research papers in the proceedings of well-known international conferences:

1. T. Moldovan, **C. Vidrighin**, I. Giurgiu, R. Potolea, "Evidence Combination for Baseline Accuracy Determination", *Proceedings of the 3rd IEEE International Conference on Intelligent computer Communication and Processing*, pp. 41-48, 2007
2. Bărbănţan, I., **Vidrighin, C.**, Borca, R., "An Offline System for Handwritten Signature Recognition", *Proceedings of Proceedings of the 5th IEEE International Conference on Intelligent computer Communication and Processing*, Cluj-Napoca, Romania, pp. 3-10, 2009
3. Cacoveanu, S., **Vidrighin, B.C.,** Potolea, R.,"Evolutional meta-learning framework for automatic classifier selection", *Proceedings of Proceedings of the 5th IEEE International Conference on Intelligent computer Communication and Processing*, pp. 27-30, 2009
4. Firte, A.A., **Vidrighin**, **B.C**., Cenan, C., "Intelligent component for adaptive E-learning systems", *Proceedings of Proceedings of the 5th IEEE International Conference on Intelligent computer Communication and Processing*, Cluj-Napoca, Romania, pp. 35-38, 2009
5. Merk, A.B., **Vidrighin**, **B.C.,** Potolea, R., "Meta-learning enhancements by data partitioning", *Proceedings of Proceedings of the 5th IEEE International*

Conference on Intelligent computer Communication and Processing, Cluj-Napoca, Romania, pp. 59-62, 2009

6. Trif, F., **Lemnaru, C.,** Potolea, R., "Identifying the User Typology for Adaptive E-learning Systems", *Proceedings of AQTR 2010 - IEEE International Conference on Automation, Quality and Testing, Robotics*, Cluj-Napoca, May 28-30, 2010, pp. 192-198/Tome III
7. Balla-Muller, N., **Lemnaru, C.,** Potolea, R., "Semi-supervised learning with lexical knowledge for opinion mining," *Proceedings of Proceedings of the 6th IEEE International Conference on Intelligent computer Communication and Processing*, Cluj-Napoca, Romania, pp.19-25, 2010
8. Bărbănţan, I., **Lemnaru, C.,** Potolea, R., "A Hierarchical Handwritten Offline Signature Recognition System", *Proceedings of the 12th International Conference on Enterprise Information Systems*, Funchal, Madeira, Portugal, pp. 139-147, 2010
9. Cacoveanu, S., **Vidrighin, B. C.,** Potolea, R., "Evaluating Prediction Strategies in an Enhanced Meta-learning Framework", *Proceedings of the 12th International Conference on Enterprise Information Systems*, Funchal, Madeira, Portugal, pp. 148-156, 2010
10. Firte, L., **Lemnaru, C.,** Potolea, R., "Spam detection filter using KNN algorithm and resampling", *Proceedings of Proceedings of the 6th IEEE International Conference on Intelligent computer Communication and Processing*, Cluj-Napoca, Romania, pp.27-33, 2010
11. Halalai, R., **Lemnaru, C.**, Potolea, R., "Distributed community detection in social networks with genetic algorithms", *Proceedings of Proceedings of the 6th IEEE International Conference on Intelligent computer Communication and Processing*, Cluj-Napoca, Romania, pp.35-41, 2010
12. R.F. Muresan, **C. Lemnaru**, R. Potolea, "Evidence Combination for Baseline Accuracy Determination", *Proceedings of Proceedings of the 6th IEEE International Conference on Intelligent computer Communication and Processing*, Cluj-Napoca, Romania, pp. 11-18, 2010
13. Dodut, A.A**., Lemnaru C**. and Potolea R., "The parallel classification of very large collections of data on multi-core platforms", in *Proceedings of the 10th International Symposium in Parallel and Distributed Computing*, pp.57-62, 2011
14. Feier, M., **Lemnaru C.** and Potolea R., "Solving NP-Complete Problems on the CUDA Architecture using Genetic Algorithms", in *Proceedings of the 10th International Symposium in Parallel and Distributed Computing*, pp.278-281, 2011
15. **Lemnaru, C.**, Firte, A., Potolea, R., "Static and Dynamic User Type Identification in Adaptive E-learning with Unsupervised Methods", *Proceedings of ICCP 2011*, pp. 11-18, 2011
16. **Lemnaru, C.**, Cuibus, M., Alic, A., Bona, A. and Potolea, R., "A Distributed Methodology for Imbalanced Classification Problems", presented at the *11th International Symposium on Parallel and Distributed Computing*, Munich, June 2012
17. **Lemnaru, C.,** Sin-Neamtiu, A., Veres, M.A., Veres, M and Potolea, R., "A System for Historical Documents Transcription based on Hierarchical Classification and Dictionary Matching", accepted at *KDIR 2012*
18. **Lemnaru, C.,** Tudose-Vintila, A., Coclici, A. and Potolea, R., "A Hybrid Solution for Imbalanced Classification Problems. Case Study on Network Intrusion Detection", accepted at *KDIR 2012*

9 Contributions and Conclusions

This thesis presents a systematic analysis of several steps involved in a data mining process, providing both theoretical and practical contributions. The general context is that of labeling a potentially large volume of noisy data, (possibly) containing irrelevant and/or redundant pieces of information; in many real-world applications, in the available data the distribution of instances belonging to each class is non-uniform, meaning that one class contains fewer instances in comparison with the others. However, the correct identification of underrepresented classes is generally of increased importance, which results in different errors possessing different degrees of gravity. Consequently, the correct choice of the metric to assess, and ultimately improve during the data mining process, is difficult – since improving one metric generally degrades others. Also, application domains may impose specific constraints on the data mining process flow.

The following specific data and goal-related issues have been addressed:

→ **Missing data**: the available data is generally incomplete. The majority of classification techniques have not been designed to deal with missing values, and often employ basic and inefficient approaches to deal with such issues. Missing data techniques are available in literature, mainly as pre-processing methods; their success in a certain scenario depends on several factors. Also, most imputation methods are oblivious to the relation between the incomplete attribute(s) and the target attribute. This thesis presents a structured analysis of existing methods (with their advantages and disadvantages – section 3.2). Also, an original global imputation method, based on non-missing data is proposed (section 3.3). The strongest assumptions made by the method are related to the existence of correlations between the incomplete and the complete attributes (including the class). The particular technique employed for learning the model used to impute each incomplete attribute – an artificial neural network ensemble – reduces the risk of imputing wrong values. Experimental results have indicated a strong correlation between the success of the imputation method on a certain attribute and the prediction power of that attribute with respect to the class. Consequently, an original joint pre-processing methodology has been introduced (section 5.1) which proposes an information exchange between data imputation and feature selection: only the attributes which are predictive for the attribute being imputed are used to build the imputation model for that attribute, and only the attributes which are predictive for the class should be imputed. The method has been currently evaluated for MCAR univariate incompleteness patterns, yielding significant performance improvements when compared to the un-processed dataset. It can be extended to accommodate random multivariate patterns as well, as long as a complete data kernel can be extracted from the data.

→ **Irrelevant/redundant data**: feature extraction is a highly laborious and domain dependent task, resulting in situations in which not all attributes recorded are relevant for the classification task, while some attributes convey redundant information. Irrelevant/redundant attributes have been shown to harm the classification process. Various feature selection techniques which tackle this issue exist in literature, with wrappers providing the most promising strategy for performance improvement; however, there is no method which is guaranteed to always produce the best possible feature subset for the given problem and selected classification method. Moreover, some method may perform significantly better than others on one particular problem, and achieve no/low performance improvement on a different problem. Also, according to the present knowledge, although the literature offers several wrapper

evaluation studies, they are not focused on comparing the performance of different wrapper combinations or analyzing how feature selection affects the choice of classification method. This thesis provides such an analysis (section 4.4.1); the results have shown that, generally, the ranking of classifiers is not significantly altered by feature selection; also, combining different classifiers for wrapper feature selection and learning is not beneficial. An original subset combination method for improving the stability of feature selection across different problems and providing an assessment of the baseline performance of feature selection in a new problem has been proposed in section 4.3 and evaluated in section 4.4.2. The method selects the most appropriate attributes by applying a global selection strategy on the attribute subsets selected individually by the search methods. Its evaluation has confirmed the fact that the method achieves better stability than individual feature selection performed via different search methods, while reducing the number of attributes significantly.

→ **Imbalanced data**: generally, in a real data mining application setting, interest cases are more difficult to collect, resulting in the interest class(es) containing fewer instances in comparison with the other classes. Although – theoretically – the class imbalance problem should affect learning algorithms, there are few systematic studies in literature which study this aspect. The present thesis provides an extensive systematic analysis on the effect of the class imbalance problem on the performance of different categories of classifiers, proposing a novel dataset characterization measure (section 7.1.3). The study revealed that all traditional algorithms are affected to some extent by the class imbalance problem, with the Multilayer Perceptron being the most robust. As opposed to the results yielded by earlier studies on artificial data, imbalance-related factors proved to have a significant impact on the performance of the Support Vector Machines. This behavior comes as a side effect appearing in imbalanced problems: the appropriate support vectors are not present in the available data. The original meta-feature proposed – the IAR, which aggregates dataset size and complexity information, has proved to offer, in conjunction with the IR, a correct indication on the expected performance of classifiers in an imbalanced problem. The present thesis presents also a thorough study on the existing approaches for dealing with imbalanced problems (section 7.2). Some methods require significant experience to produce improvements, while others are restricted to a specific category of classifiers. To address these issues, an original meta-classification strategy, has been proposed – Evolutionary Cost-Sensitive Balancing (section 7.3). The method performs a genetic search in order to simultaneously tune the base classifier's parameters and identify the most appropriate cost matrix, which is then employed by a cost-sensitive meta-learner, in conjunction with the base classifier. An advantage of the method, besides its generality, is the fact that the cost matrix is determined indirectly. Thus, the difficult problem of setting appropriate error costs is translated into selecting the appropriate fitness measure, given the specific problem goals. Comparative empirical evaluations on benchmark data have proved that ECSB significantly improves the performance of the base classifiers in imbalanced conditions, achieving superior results to sampling with SMOTE or adapting the algorithm to the imbalance via evolutionary parameter selection; also, ECSB achieves superior results to current prominent approaches in literature: Evolutionary Under-Sampling and a complex Support Vector Machines ensemble; the most successful cost-sensitive strategy is predicting the class with minimum expected misclassification cost, instead of the most likely class; balanced metrics are generally appropriate as fitness functions; for extreme problems, such as those in which

precision is of utmost importance, or recall is the only important measure, imbalanced metrics – such as parameterized combinations between primary metrics, are more suitable.

→ **Imbalanced error costs**: classification errors may possess different degrees of gravity, according to the domain specifics. Such situations are covered by cost-sensitive learning, with a series of algorithms existing in literature; the present thesis presents a structured analysis of prominent cost-sensitive methods (section 6.2). Most algorithms focus on a single type of cost, whereas real-world scenarios frequently include two types of costs: test and error costs. The best known method which considers simultaneously both types of costs is the ICET algorithm. The main contributions presented in this area focus on a series of improvements to the original ICET algorithm, as well as filling in the gap in the algorithm evaluation part (section 6.3). Consequently, a new version of the algorithm has been developed, ProICET, which follows the basic ICET technique, while improving the robustness of the genetic component, by: the employment of a single population technique and elitism, with rank-based fitness assignment for parent selection, and increasing search variability via the recombination operators. A series of evaluations focused on assessing the performance of ProICET under a series of different aspects have been performed: a systematic empirical analysis on the effectiveness of stratification in reducing cost asymmetries and a comparative empirical study on the misclassification and total costs. The results indicate that it provides a robust approach, achieving lowest total costs. Also, ProICET has been applied to a real problem related to prostate cancer diagnosis and prognosis (within the CEEX 18/2005 IntelPRO research grant): predicting the value of post-operative PSA for patients who have undergone prostate surgery, from data recorded pre- and immediately after the operation, confirming its robustness.

→ **Classifier and metric selection:** There is no universally best classifier which performs better than all the others on every possible problem, given any evaluation metric. Moreover, no general rules which indicate the appropriate metric to select in a certain context exist, although there are some application domains which benefit from specialized metrics. Translating data characteristics and the ultimate goal of the classification process into the appropriate performance metric, selecting the most appropriate classifier given the data and goal characteristics, finding the best parameter settings for the classifier are therefore essential points in achieving a successful data mining process. The current thesis presents an analysis of the most important performance metrics (section 2.3); also, throughout the thesis – but mainly in Chapters 6 and 7 – the suitability of different measures for special classification scenarios is analyzed. Section 8.1 presents an original meta-learning framework for automated classifier selection, together with a method for baseline performance assessment. Several original design elements have been introduced into the framework, the most important being a wide selection of dataset meta-features, several neighbor selection strategies, a general-purpose aggregate metric and the possibility to output a ranking of classifiers, rather than a single candidate. Experimental results on benchmark datasets have made possible the identification of a best prediction strategy among several investigated.

→ **Data mining process instantiations:** Application domains may impose specific constraints on the data mining process, such as having an interpretable classification model, or a reasonable training time, the capacity to perform classification on a large number of classes, each having a limited amount of training instances. Several such

applicative issues are tackled and original contributions (both theoretical and practical) are presented (sections 8.2-8.4).

The main original contributions presented throughout this thesis can be grouped into:

1. *General DM contributions - theoretical:*
    - a systematic analysis of existing issues and solutions for: handling incomplete data, feature selection, imbalanced classification, cost-sensitive classification, classification algorithms and performance assessment strategies and metrics
    - a global imputation method, based on non-missing data
    - a combination method for wrapper search strategies, to improve selection stability
    - an original joint pre-processing strategy, which employs feature selection information in the data imputation step
    - proposal of a new meta-feature – IAR – for characterizing a dataset with respect to the imbalance-related factors; the new meta-feature, in conjunction with the IR, has been proven to offer correct indication on the expected performance of classifiers in imbalanced problems
    - a general method for imbalanced classification – ECSB – a meta-classification strategy, which can be applied to any error-reduction classifier
    - a cascade combination method for classifier baseline performance assessment, with improved stability across different domains
    - a composite metric for general classifier performance assessment

2. *General DM contributions – practical:*
    - a systematic empirical study on wrapper combinations for feature selection, to study possibility of combining different classifiers for the steps of feature selection and learning, the effect of pruning on feature selection and how feature selection affects, generally, the choice of the learning scheme
    - enhancements on the ICET cost-sensitive classifier. The new version of the algorithm, ProICET, achieves superior performance to the original algorithm and to other known cost-sensitive algorithms in literature in terms of total cost and accuracy
    - a systematic analysis on the effect of the class-imbalance problem on the performance of different types of classifiers, considering a large number of benchmark problems, several imbalance-related factors and performance metrics and six different classification techniques
    - a meta-learning framework for automated classifier selection, which employs various meta-features and several different prediction strategies, while reducing user involvement at a minimum
    - a distributed version for the original ECSB method from Chapter 7
    - a hierarchical classifier for problems having a large number of classes, based on clustering and classification sub-models
    - a hierarchical model for classification of multi-class imbalanced problems, which consists of a multiple classifier on a first level, and a level 2 classifier focused on labeling difficult cases

- a classifier based on data partitioning
- a parallel version for the SPRINT decision tree classifier
- a lightweight GPGPU parallel genetic framework

3. *Domain specific contributions – theoretical:*
   - a dynamic composite fitness function for the community detection problem
   - a model for static and dynamic user-type identification in adaptive e-learning systems

4. *Domain specific contributions – practical:*
   - a system for offline signature recognition
   - a system for network intrusion detection
   - a system for historical documents transcription based on hierarchical classification and dictionary matching
   - a system for distributed community detection, using genetic algorithms
   - a system for opinion mining, using semi-supervised learning with lexical knowledge
   - a spam detection filter, with learned training distribution

All newly proposed classification methods have been empirically evaluated and compared to other prominent approaches in literature. For pre-processing methods, their effect on the overall performance of a classification task has been studied.

The results of the research efforts have been partially applied in three research grants, between 2007 and 2011. Also, they have been disseminated within the scientific community, through the publication of:

- 4 journal papers – 1 as first author – out of which 3 published in B+ journals
- 1 paper submitted at the Journal of Data Mining and Knowledge Discovery (ISI indexed) – review requested
- 4 book chapters – 2 as first author – out of which 3 published in Springer *Lecture Notes in Business Information Processing* series
- 30 research papers – 12 as first author – in the proceedings of renowned international conferences (indexed by IEEEXplore – 19, ISIPro – 14, DBLP – 12 and CSDL – 7)
- 8 independent citations – out of which 4 in ISI-indexed journals

The current research efforts focus on adding an extra genetic algorithms tuning level to the dECSB method, and validating the strategy for larger datasets as well as for multi-class real-world imbalanced problems. In addition, the joint pre-processing methodology has to be extended to accommodate successfully any incompleteness mechanism and pattern.

The original methods presented in this work have been successfully validated on a number of benchmark real-world datasets – well known and intensely employed by the scientific community, which originate from real-world scenarios. The ProICET enhanced cost-sensitive method has also been deployed successfully in a real-world medical diagnosis problem. I believe that the original pre-processing methods and the methodology for imbalanced classification will improve the quality of the DM classification process in real world settings displaying the issues addressed in the current thesis.

References

1. [Agr96] Agrawal R., Shafer J.C., Mehta M. (1996). SPRINT: A Scalable Parallel Classifier for Data Mining. Proceedings of the 22nd International Conference on Very Large Data Bases. pp. 544-555

2. [Ala08] Alaiz-Rodríguez R. and Japkowicz N. (2008). Assessing the impact of changing environments on classifier performance. In Canadian AI'08: Proceedings of the Canadian society for computational studies of intelligence, 21st conference on advances in artificial intelligence (pp. 13–24). Berlin, Heidelberg: SpringerVerlag

3. [Ali06] Aliamiri A. (2006). *Statistical Methods for Unexploded Ordnance Discrimination*. PhD Thesis. Department of Electrical and Computer Engineering. Northeastern University. Boston MA

4. [All09] Allison, P.D. (2009). Missing Data. The SAGE Handbook of Quantitative Methods in Psychology, edited by Roger E. Millsap and Alberto Maydeu-Olivares. Thousand Oaks, CA: Sage Publications Inc. pp. 72-89

5. [Alm08] Almohri H., Gray J.S., Alnajjar H. (2008). *A Real-time DSP-Based Optical Character Recognition System for Isolated Arabic characters using the TI TMS320C6416T*. PhD Thesis.

6. [Alm97] Almuallim H., Dietterich T.G. (1997). Learning with many irrelevant features. In Proceedings of Ninth National Conference on AI, pp. 547-552.

7. [Bac06] Bach F.R., Heckerman D., and Horvitz E. (2006). Considering cost asymmetry in learning classifiers. The Journal of Machine Learning Research, 7:1713–1741. [Bac06] Bach F.R., Heckerman D., and Horvitz E. (2006). Considering cost asymmetry in learning classifiers. The Journal of Machine Learning Research, 7:1713–1741.

8. [Bac91] Bäck, T. and Hoffmeister, F. (1991). Extended Selection Mechanisms in Genetic Algorithms. In Proceedings of the Fourth International Conference on Genetic Algorithms, San Mateo, California, USA: Morgan Kaufmann Publishers, pp. 92-99

9. **[Bal10]** Balla-Muller N., **Lemnaru C.** and Potolea R. (2010). Semi-supervised learning with lexical knowledge for opinion mining. Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing. pp.19-25

10. [Bar03] Barandela R., Sanchez J.S., Garcia V. and Rangel E. (2003). Strategies for learning in class imbalance problems. Pattern Recognition. 36(3): 849–851.

11. [Bat03] Batista G.E.A.P.A. and Monard M.C. (2003). An analysis of four missing data treatment methods for supervised learning. Applied Artificial Intelligence, vol. 17, no. 5-6, pp. 519-533.

12. **[Băr09]** Bărbănţan I., **Lemnaru C.** and Borca R. (2010). An Offline System for Handwritten Signature Recognition. Proceedings of the 5th IEEE ICCP, Cluj-Napoca, pp. 3-10.

13. **[Băr10**] Bărbănţan I., **Lemnaru C.** and Potolea R. (2010). A Hierarchical Handwritten Offline Signature Recognition System. Proceedings of the 12th International Conference on Enterprise Information Systems, Funchal, Madeira, Portugal. pp. 139-147

14. [Ben00] Bensusan H., Giraud-Carrier C. and Kennedy C.J. (2000). A Higher-Order Approach to Meta-Learning. Proceedings of the ECML-2000 Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination. pp. 33-42

15. [Bha08] Bhatnagar V. and Gupta S.K. (2008). Modeling the KDD Process. Encyclopedia of Data Warehousing and Mining, Second Edition. Information Science Reference, Hershey, New York, pp. 1337 – 1344

16. [Bre11] Brehar R. and Nedevschi S. (2011). A comparative study of pedestrian detection methods using classical Haar and HoG features versus bag of words model computed from Haar and HoG features. Proceedings of IEEE Intelligent Computer Communication and Processing, pp. 299–306.

17. [Bre84] Breiman, L., Friedman J. H., Olshen R. A. and Stone C.J. (1984). Classification and regression trees. Monterey, Calif., U.S.A.: Wadsworth, Inc

18. [Bre96] Breiman L. (1996). Bagging predictors. Machine Learning, 24, pp. 123-140

19. [Bro10] Brodersen K.H., Ong C.S., Stephen K.E., and Buhmann J.M. (2010) The balanced accuracy and its posterior distribution. Proc. of the 20th Int. Conf. on Pattern Recognition. pp. 3121–3124

20. **[Cac09]** Cacoveanu S., **Vidrighin B.C.** and Potolea R. (2009). Evolutional meta-learning framework for automatic classifier selection. Proceedings of the 5th IEEE ICCP, pp. 27-30.

21. **[Cac10]** Cacoveanu S., **Vidrighin B.C.** and Potolea R. (2010). Evaluating prediction strategies in an enhanced meta-learning framework. Proceedings of the 12th International Conference on Enterprise Information Systems, Funchal, Madeira, Portugal, pp. 148-156.

22. [Cha04] Chai X., Deng L., Yang Q., Ling C.X. (2004). Test-Cost Sensitive Naive Bayes Classification. Proceedings of the 4th IEEE International Conference on Data Mining (ICDM'04*)*, p.51-58.

23. [Cha96] Chan P. (1996). An Extensible Meta-Learning Approach for Scalable and Accurate Inductive Learning. PhD thesis. Columbia University.

24. [Cha98] Chan P. and Stolfo S. (1998). Toward scalable learning with non-uniform class and cost distributions: a case study in credit card fraud detection. In *Proceedings*

of the Fourth International Conference on Knowledge Discovery and Data Mining, AAAI Press, pp. 164-168.

25. [Cha02] Chawla N.V., Bowyer K.W., Hall L.O., Kegelmeyer W.P. (2002). SMOTE: Synthetic Minority Over-Sampling Technique. *Journal of Artificial Intelligence Research*. 16:321—357.

26. [Cha03] Chawla N.V., Lazarevic A., Hall L.O. and Bowyer K.W. (2003). SMOTEboost: Improving prediction of the minority class in boosting. In *Proceedings of the Seventh European Conference on Principles and Practice of Knowledge Discovery in Databases*. 107—119.

27. [Cha04] Chawla N.V., Japkowicz N., Kotcz A. (2004). Editorial: special issue on learning from imbalanced datasets. *SIGKDD Explorations*. 6(1):1–6.

28. [Cha06] Chawla N. (2006). Data Mining from Imbalanced Datasets: An Overview. *Data Mining and Knowledge Discovery Handbook*. Springer US. 853—867.

29. [Che96] Cherkauer K. J. and Shavlik J. W. (1996). Growing simpler decision trees to facilitate knowledge discovery. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. AAAI Press.

30. [Das97] Dash M. and Liu H. (1997). Feature Selection for Classification. *Intelligent Data Analysis 1*, INSTICC Press. 131–156.

31. [Den10] Denil, M. and Trappenberg, T. (2010). Overlap versus imbalance. In Canadian AI 2010, LNAI, Vol. 6085, pp. 220–231.

32. [Der02] Derderian K. (2002). General Genetic Algorithm Tool, http://www.karnig.co.uk/ga/ggat.html, last accessed Nov. 2011

33. [Dev82] Devijver P.A. and Kittler J. (1982). *Pattern Recognition – A Statistical Approach*. Prentice Hall, London, GB.

34. **[Dod11]** Doduţ A.A.**, Lemnaru C.** and Potolea R. (2011). The parallel classification of very large collections of data on multi-core platforms. In Proceedings of ISPDC 2011, pp. 57-62.

35. [Dom97] Domingos, P. (1997). Context-sensitive feature selection for lazy learners. *Artificial Intelligence Review*, (11):227–253.

36. [Dom99] Domingos P. (1999). MetaCost: A General Method for Making Classifiers Cost-Sensitive. *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, ACM Press, 155-164.

37. [Dra94] Draper, B., Brodley, C. and Utgoff, P. (1994). Goal-directed classification using linear machine decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):888-893.

38. [Elk00] Elkan C. (2000). Results of the KDD'99 classifier learning. *SIGKDD Explorations*, vol.1, no.2, pp. 63-64.

39. [Est01] Estabrooks A. and Japkowicz N. (2001). A mixture-of-experts framework for text classification. *Proceedings of the 2001 workshop on Computational Natural Language Learning* - Volume 7. Toulouse, France. pp. 34-43.

40. [Fan00] Fan W., Stolfo S., Zhang J. and Chan P. (2000). AdaCost: Misclassification cost-sensitive boosting. *Proceedings of the 16th International Conference on Machine Learning*, pp. 97–105.

41. [Far09] Farid D.M., Darmont J., Harbi N., Hoa N.H., Rahman M.Z. (2009). Adaptive Network Intrusion Detection Learning: Attribute Selection and Classification. *World Academy of Science, Engineering and Technology* 60.

42. [Faw97] Fawcett T. and Provost F.J. (1997). Adaptive Fraud Detection. *Data Mining and Knowledge Discovery*, 1(3), pp. 291-316

43. [Faw06] Fawcett T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27, 861–874.

44. [Fay96] Fayyad U.M., Piatetsky-Shapiro G. and Smyth P. (1996). From Data Mining to Knowledge Discovery in Databases. *Artificial Intelligence Magazine*, 17(3): 37-54.

45. **[Fei11]** Feier M., **Lemnaru C.** and Potolea R. (2011). Solving NP-Complete Problems on the CUDA Architecture using Genetic Algorithms. In *Proceedings of ISPDC 2011*, pp. 278-281,

46. **[Fir09]** Firte A.A., **Vidrighin B.C.** and Cenan C. (2009). Intelligent component for adaptive E-learning systems. *Proceedings of the IEEE 5th International Conference on Intelligent Computer Communication and Processing*. 27-29 August 2009, Cluj-Napoca, Romania. pp. 35-38.

47. **[Fir10]** Firte L., **Lemnaru C.** and Potolea R. (2010). Spam detection filter using KNN algorithm and resampling. *Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing*, pp.27-33.

48. [Fre97] Freund Y. and Shapire R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

49. [Gar09] García S. and Herrera F. (2009). Evolutionary Undersampling for Classification with Imbalanced Datasets: Proposals and Taxonomy. *Evolutionary Computation*, Vol. 17, No. 3. pp. 275-306.

50. [Ged03] Gediga G. and Duntsch I. (2003). Maximum consistency of incomplete data via noninvasive imputation. *Artificial intelligence Review*, vol. 19, pp. 93-107.

51. [Gen89] Gennari, J.H., Langley P. and Fisher D. (1989). Models of incremental concept formation. *Artificial Intelligence*, 40, pp.11-61.

52. [Gog10] Gogoi P., Borah B., Bhattacharyya D.K., (2010). Anomaly Detection Analysis of Intrusion Data using Supervised & Unsupervised Approach. *Journal of Convergence Information Technology*, vol. 5, no. 1, pp. 95-110

53. [Gre86] Grefenstette, J.J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16, 122-128.

54. [Grz02] Grzymala-Busse J.W., Grzymala-Busse W.J. and Goodwin L.K. (2002). A comparison of three closest fit approaches to missing attribute values in preterm birth data. *International journal of intelligent systems*, vol. 17, pp. 125-134

55. [Grz05] Grzymala-Busse J.W., Stefanowski J. and Wilk S. (2005). A comparison of two approaches to data mining from imbalanced data. *Journal of Intelligent Manufacturing*, 16. Springer Science+Business Media Inc. pp. 565–573

56. [Guo04] Guo H. and Viktor H.L. (2004). Learning from imbalanced datasets with boosting and data generation: The Databoost-IM approach. *SIGKDD Explorations*, 6(1), pp. 30–39.

57. [Guy02] Guyon I. Weston J., Barnhill S. and Vapnik V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, Vol. 46, pp. 389–422.

58. [Guy03] Guyon I. and Elisseeff A. (2003). An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, no. 3/2003, pp. 1157-1182.

59. **[Hal10]** Halalai R., **Lemnaru C.** and Potolea R. (2010). Distributed community detection in social networks with genetic algorithms. *Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing*, pp. 35-41.

60. [Hal00] Hall, M. (2000). *Correlation-based feature selection for machine learning*. PhD Thesis, Department of Computer Science, Waikato University, New Zealand.

61. [Han06] Hand D.J. (2006) Classifier technology and the illusion of progress (with discussion). Statistical Science, 21, 1-34

62. [Har68] Hart P.E. (1968). The Condensed Nearest Neighbor Rule. *IEEE Transactions on Information Theory* IT-14, 515—516.

63. [Hol89] Holte R.C., Acker L.E. and Porter B.W. (1989). Concept learning and the problem of small disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 813-818.

64. [Hua06] Huang K., Yang H., King I., and Lyu M.R. (2006). Imbalanced Learning with a Biased Minimax Probability Machine. *IEEE Transactions on Systems, Man, and Cybernetics*, Part B: Cybernetics, 36(4): 913—923.

65. [Jap02] Japkowicz N. and Stephen S. (2002). The Class Imbalance Problem: A Systematic Study. *Intelligent Data Analysis Journal*, Volume 6, Number 5. pp. 429—449.

66. [Joh94] John G.H., Kohavi R. and Pfleger P. (1994). Irrelevant features and the subset selection problem. In *Machine Learning: Proceedings of the Eleventh International Conference*. Morgan Kaufmann.

67. [Kar98] Karakoulas G. and Shawe-Taylor J. (1998). Optimizing classifiers for imbalanced training sets. In: *Proceedings of Neural Information Processing Workshop*, NIPS'98, pp. 253-259.

68. [Kir92] Kira, K., Rendell, L. A., The feature selection problem - Traditional methods and a new algorithm. In Proceedings of Ninth National Conference on AI, pp. 129-134.

69. [Koh95] Kohavi, R. (1995). Wrappers for Performance Enhancement and Oblivious Decision Graphs. PhD thesis, Stanford University, Computer Science Department

70. [Koh97] Kohavi R. and John J.H. (1997). Wrappers for feature subset selection. Artificial Intelligence, Volume 7, Issue 1-2.

71. [Kol06] Koljonen J. and Alander J.T. (2006). Effects of population size and relative elitism on optimization speed and reliability of genetic algorithms. In Proceedings of the Ninth Scandinavian Conference on Artificial Intelligence (SCAI 2006), Honkela, Kortela, Raiko, Valpola (eds.), pp. 54–60

72. [Kon94] Kononenko, I. (1994). Estimating attributes: analysis and extensions of Relief. In De Raedt, L. and Bergadano, F., editors, Machine Learning: ECML-94, pages 171-182. Springer Verlag.

73. [Kub97] Kubat M. and Matwin S. (1997). Addressing the Course of Imbalanced Training Sets: One-sided Selection. In *ICML*. 179—186

74. [Lan94a] Langley P. and Sage S. (1994). Induction of selective Bayesian classifiers. In Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence, Seattle, W.A, Morgan Kaufmann, pp. 399-406

75. [Lan94b] Langley P. and Sage, S. (1994). Scaling to domains with irrelevant features. In R. Greiner, editor, Computational Learning Theory and Natural Learning Systems, volume 4. MIT Press.

76. [Lau01] Laurikkala J. (2001). Improving Identification of Difficult Small Classes by Balancing Class Distribution. Technical Report A-2001-2, University of Tampere.

77. **[Lem11a] Lemnaru C**, Firte A. and Potolea R. (2011). Static and Dynamic User Type Identification in Adaptive E-learning with Unsupervised Methods. Proceedings of 7th ICCP, pp. 11-18.

78. **[Lem11b] Lemnaru C.** and Potolea R. (2011). Imbalanced Classification Problems: Systematic Study, Issues and Best Practices. To appear in Lecture Notes in Business Intelligence, Springer-Verlag.

79. **[Lem12a] Lemnaru C.**, Cuibus M., Bona A., Alic A. and Potolea R. (2012). A Distributed Methodology for Imbalanced Classification Problems, presented at the 11th International Symposium on Parallel and Distributed Computing, Munich, June 2012.

80. **[Lem12b] Lemnaru C.**, Sin-Neamtiu A., Veres M.A. and Potolea R. (2012). A System for Historical Documents Transcription based on Hierarchical Classification and Dictionary Matching, accepted at KDIR 2012.

81. **[Lem12c] Lemnaru C.,** Tudose-Vintila A., Coclici A. and Potolea R. (2012). A Hybrid Solution for Imbalanced Classification Problems – Case Study on Network Intrusion Detection, accepted at KDIR 2012.

82. [Lin02] Lin Y., Lee Y., Wahba G. (2002). Support vector machines for classification in nonstandard situations, Mach. Learn. 46. 191–202

83. [Lin04] Ling C.X., Yang Q., Wang J. and Zhang S. (2004). Decision trees with minimal costs. *ACM International Conference Proceeding Series: 21st International Conference on Machine Learning*. C.E. Brodley (ed.), New York, USA: ACM Press Article No. 69, pp. 4–8.

84. [Liu96] Liu H., Setiono R. (1996). A probabilistic approach to feature selection–a filter solution. In *Proceedings of International Conference on Machine Learning*, pp. 319-327.

85. [Liu00] Liu B., Ma Y., Wong C.K. (2000). Improving an association rule based classifier. *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*. pp. 504–509

86. [Liu10] Liu W., Chawla S., Cieslak D., Chawla N. (2010). A Robust Decision Tree Algorithm for Imbalanced Datasets. In: *Proceedings of the Tenth SIAM International Conference on Data Mining*, 766–777

87. [Liu11] Liu W. and Chawla S., (2011). Class Confidence Weighted kNN Algorithms for Imbalanced Datasets. *Advances in Knowledge Discovery and Data Mining*, LNCS, Volume 6635/2011. 345-356.

88. [Mag04] Magnani, M. (2004). Techniques for Dealing with Missing Data in Knowledge Discovery Tasks, (available at http://magnanim.web.cs.unibo.it/index.html)

89. [Mar00] Margineantu D.D. (2000). When does imbalanced data require more than cost-sensitive learning? In Workshop on Learning from Imbalanced Data, Artificial Intelligence (AAAI-2000).

90. [Mar03] Margineantu D. and Dietterich T. (2003). A wrapper method for cost-sensitive learning via stratification. Available at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.27.1102 (Accessed Nov 2011)

91. [Mas10] Masnadi-Shirazi H. and Vasconcelos N. (2010). Risk minimization, probability elicitation, and cost-sensitive SVMs. In *Proceedings of the International Conference on Machine Learning*, pp. 204-213. ACM Press.

92. [Meh96] Mehta M., Agrawal R. and Rissanen J. (1996). SLIQ: A fast scalable classifier for data mining. In *Proc. of the 5th Intl. Conference on Extending Database Technology (EDBT)*, Avignon, France. pp. 18-32.

93. **[Mer09]** Merk A.B., **Vidrighin B.C.,** Potolea R. (2009). Meta-learning enhancements by data partitioning. Proceedings of the 5th IEEE ICCP, pp. 59-62.

94. [Min01] Minert R.P. (2001). Deciphering Handwriting in German Documents: Analyzing German, Latin and French in Vital Records Written in Germany. GRT Publications.

95. **[Mol07]** Moldovan T., Vidrighin C., Giurgiu I., Potolea R. (2007). Evidence Combination for Baseline Accuracy Determination. Proceedings of the 3rd ICCP 2007, 6-8 September, Cluj-Napoca, Romania, pp. 41-48.

96. [Mol02] Molina L.C., Belanche L., Nebot A. (2002). Feature Selection Algorithms: A Survey and Experimental Evaluation. In Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02), p.306.

97. [Moo94] Moore A.W. and Lee M.S. (1994). Efficient algorithms for minimizing cross validation error. In Machine Learning: Proceedings of the Eleventh International Conference. Morgan Kaufmann.

98. **[Mur10]** Muresan R.F., **Lemnaru C.** and Potolea R. (2010). Evidence Combination for Baseline Accuracy Determination. Proceedings of the 6th ICCP, pp. 11-18.

99. [Ned10] Nedevschi S., Peter I.R., Dobos, I.A. and Prodan C. (2010). An improved PCA type algorithm applied in face recognition. Proceedings of 2010 IEEE Intelligent Computer Communication and Processing, (ICCP 2010), August 26-28, pp. 259-262.

100. [Ned09] Nedevschi S., Bota S. and Tomiuc C. (2009). Stereo-Based Pedestrian Detection for Collision-Avoidance Applications. IEEE Transactions on Intelligent Transportation Systems, vol. 10, no. 3, pp. 380-391.

101. [Ngu08] Nguyen H.A., Choi D. (2008). Application of Data Mining to Network Intrusion Detection: Classifier Selection Model, APNOMS, LNCS 5297, pp. 399–408.

102. [Nil07] Nilsson R. (2007). Statistical Feature Selection, with Applications in Life Science. PhD Thesis. Linkoping University.

103. [Nor89] Norton S.W. (1989). Generating better decision trees. Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, IJCAI-89, pp 800-805.

104. [Nun91] Nunez M. (1991). The Use of Background Knowledge in Decision Tree Induction. Machine Learning, 6, 231-250.

105. **[Ona07]** Onaci A., **Vidrighin C.**, Cuibus M. and Potolea R. (2007). Enhancing Classifiers through Neural Network Ensembles. Proceedings of the 3rd ICCP 2007, 6-8 September, Cluj-Napoca, Romania, pp. 57-64.

106. [Par98] Park Y. and Song M.A. (1998). Genetic Algorithm for Clustering Problems. In Genetic Programming 1998: Proceedings of the Third Annual Conference, pp. 568–575.

107. [Paz94] Pazzani M., Merz C., Murphy P., Ali K., Hume T. and Brunk C. (1994). Reducing misclassification costs. Proceedings of the 11th International Conference on Machine Learning, USA: Morgan Kaufmann, 217–225.

108. [Paz95] Pazzani, M. (1995). Searching for dependencies in Bayesian classifiers. In Proceedings of the Fifth International Workshop on AI and Statistics.

109. [Pen03] Pendharkar P.C., Nanda S., Rodger J.A. and Bhaskar R. (2003). An Evolutionary Misclassification Cost Minimization Approach for Medical Diagnosis. In P. Pendharkar (Ed.), Managing Data Mining Technologies in Organizations: Techniques and Applications, pp. 32-44.

110. [Piz98] Pizzuti C. (2008). GA-Net: A genetic algorithm for community detection in social networks. Lecture Notes in Computer Science, vol. 5199. pp. 1081-1090.

111. **[Pot11a]** Potolea R., Bărbănţan I. and **Lemnaru C.** (2011). A Hierarchical Approach for the Offline Handwritten Signature Recognition. Lecture Notes in Business Information Processing, Volume 73, Part 3, 264-279.

112. [P**ot11b**] Potolea R., Cacoveanu S. and **Lemnaru C.** (2011). Meta-learning Framework for Prediction Strategy Evaluation. Lecture Notes in Business Information Processing, Volume 73, Part 3, 280-295.

113. **[Pot11c]** Potolea R., **Lemnaru C.** (2011). A Comprehensive Study of the Effect of Class Imbalance on the Performance of Classifiers. Proceedings ICEIS 2011, pp. 14-21.

114. [Pro97] Provost F. and Fawcett T. (1997). Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distributions. Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, pp. 43-48.

115. [Qui91] Quinlan J.R. (1991). Improved estimates for the accuracy of small disjuncts. Mach. Learn. 6. 93–98.

116. [Qui93] Quinlan J.R. (1993). C4.5: Programs for Machine Learning, Morgan Kaufmann.

117. [Qui96] Quinlan J.R. (1996). Boosting first-order learning. Proceedings of the 7th International Workshop on Algorithmic Learning Theory. 1160:143–155.

118. [Rub87] Rubin, D.B. (1987). Multiple imputation for non-response in surveys, John Wiley and Sons.

119. [Rub87b] Rubin D.B. and Little RJA (1987). Statistical Analysis with Missing Data, J. Wiley & Sons, New York.

120. [Sar98] Sarle, W.S. (1998). Prediction with missing inputs. Technical Report, SAS Institute Inc., SAS Campus Drive, Cary, NC 27513, USA.

121. [Sch04] Schoner H. (2004). Working with Real-World Datasets. PhD Thesis. School of Engineering , Technical University of Berlin. 176p.

122. [She06] Sheng V.S. and Ling C.X. (2006). Thresholding for Making Classifiers Cost-sensitive. In Proceedings of the 21st National Conference on Artificial Intelligence, 476-481.

123. [Sin08] Sindhwani V. and Melville P (2008). Document-Word Co-Regularization for Semi-supervised Sentiment Analysis. IEEE International Conference on Data Mining (ICDM). pp. 1025–1030.

124. [Spe91] Spears W.M. and De Jong K.A. (1991). An Analysis of Multi-Point Crossover. In Foundations of Genetic Algorithms. San Mateo, California, USA: Morgan Kaufmann Publishers, 1991, pp. 301-315.

125. [Sun07] Sun Y., Kamel M.S., Wong A.K.C., Wang Y (2007). Cost-sensitive boosting for classification of imbalanced data. Pattern Recognition, v.40 n.12, p.3358-3378.

126. [Tan89] Tan M. and Schlimmer J. (1989). Cost-Sensitive concept learning of sensor use in approach and recognition. Proceedings of the Sixth International Workshop on Machine Learning, ML-89, pp. 392-395.

127. [Tav09] Tavallaee M., Bagheri M., Wei L., Ghorbani A.A. (2009). A Detailed Analysis of the KDD CUP 99 Dataset. Proceedings of the IEEE Symposium on Computational Intelligence in Security and Defense Applications, pp. 53-58.S

128. [The97] Therneau T.M. and Atkinson E.J. (1997). An introduction to recursive partitioning using the RPART routines. Technical Report, Mayo Foundation. Department of Statistics, Stanford University, USA.

129. [Tia11] Tian J., Gu H. and Liu W. (2011). Imbalanced classification using support vector machine ensemble. Neural Comput. Appl. 20, 2:203-209.

130. [Tin98] Ting K.M. (1998). Inducing cost-sensitive decision trees via instance weighting. Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery. Berlin: Springer Verlag, 139–147.

131. [Tin02] Ting K.M. (2002). An instance-weighting method to induce cost-sensitive decision trees. IEEE Transactions on Knowledge and Data Engineering. 14, 659–665.

132. [Tom76] Tomek I. (1976). Two Modifications of CNN. IEEE Transactions on Systems Man and Communications SMC-6, 769—772.

133. [Tri10] Trif F., **Lemnaru C,.** Potolea R. (2010). Identifying the User Typology for Adaptive E-learning Systems. Proceedings of AQTR 2010 - IEEE International Conference on Automation, Quality and Testing, Robotics, Cluj-Napoca, May 28-30, 2010, pp. 192-198/Tome III.

134. [Tur95] Turney P. (1995). Cost sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. Journal of Artificial Intelligence Research, (2):369–409.

135. [Tur00] Turney P. (2000). Types of Cost in Inductive Concept Learning. In: Proceedings of the Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning, Stanford University, California, pp. 15-21.

136. [Two99] Two Crows Corp. Introduction to Data Mining and Knowledge Discovery, Third Edition, http://www.twocrows.com/intro-dm.pdf , 1999, last accessed in November, 2008

137. [UCI] UCI Machine Learning Data Repository, http://archive.ics.uci.edu/ml (last accessed on Jan. 2012)

138. [Vam09] Vamvakas G., Gatos B., Perantonis S.J. (2009). A Novel Feature Extraction and Classification Methodology for the Recognition of Historical Documents. Proc. of 10th International Conference on Document Analysis and Recognition (ICDAR'09), pp 491-495.

139. [Ver96] Vermunt J.D. (1996). Metacognitive, cognitive and affective aspects of learning styles and strategies: A phenomenographic analysis. Higher Education 31, 25–50.

140. [Vil04] Vilalta R., Giraud-Carrier C., Brazdil P., and Soares C. (2004). Using Meta-Learning to Support Data Mining. International Journal of Computer Science & Applications, pp. 31-45.

141. [Vis05] Visa S. and Ralescu A. (2005). Issues in mining imbalanced datasets-a review paper. In: Proc. of the Sixteen Midwest Artificial Intelligence and Cognitive Science Conference, 67–73.

142. [Wei03] Weiss G. and Provost F. (2003). Learning when Training Data are Costly: The Effect of Class Distribution on Tree Induction. Journal of Artificial Intelligence Research 19. pp. 315-354.

143. [Wei04] Weiss, G. (2004). Mining with Rarity: A Unifying Framework, SIGKDD Explorations 6(1), 7—19.

144. [Wil09] Williams D., Myers V., Silvious M. (2009). Mine classification with imbalanced data. IEEE Geoscience and Remote Sensing Letters. 6(3):528–532.

145. [Wit05] Witten I.H. and Frank E. (2005). Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, San Francisco, 2nd edition.

146. [Wu03] Wu G., Chang E.Y. (2003). Class-boundary alignment for imbalanced dataset learning. Proceedings of the ICML'03 Workshop on Learning from Imbalanced Datasets, pp. 49-56.

147. [Yen06] Yen S. and Lee Y. (2006). Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset. In ICIC, LNCIS 344, pages 731–740.

148. [Yoo05] Yoon K. and Kwek S. (2005). An unsupervised learning approach to resolving the data imbalanced issue in supervised learning problems in functional genomics. In HIS '05: Proceedings of the Fifth International Conference on Hybrid Intelligent Systems, pages 303–308.

149. [Zad01] Zadrozny B., Elkan C. (2001). Learning and making decisions when costs and probabilities are both unknown. In: Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining, 204–213.

150. [Zad03] Zadrozny B., Langford J. and Abe N. (2003). Cost-Sensitive Learning by Cost-Proportionate Example Weighting. ICDM '03 Proceedings of the Third IEEE International Conference on Data Mining, 435-442.

151. [Zho06] Zhou Z.H. and Liu X.Y. (2006). Training Cost-Sensitive Neural Networks with Methods Addressing the Class Imbalance Problem. IEEE Trans. on Knowl. and Data Eng. 18, 1. 63-77.

Research Grants and Publications

Research Grants

1. **IntelPRO - Intelligent System for Assisting the Therapeutic Decision for Patients with Prostate Cancer,** *National research grant funded by ANCS, CEEX - INFOSOC, (2005-2008), no. 18/2005;* http://cv.utcluj.ro/intelpro/
2. **SEArCH - Adaptive eLearning Systems using Concept Maps,** *National grant funded by CNMP Program 4: Research partnership for priority domains, (2008-2011), no. 12080/2008;* http://search.utcluj.ro/
3. **PhD Researcher grant,** *National Research grant funded by CNCSIS – BD type, 2008-2010, no.348/2008*

Journal Papers

1. Potolea R. and **Lemnaru C.**, "Evolutionary Cost-Sensitive Balancing: A Generic Method for Imbalanced Classification Problems", submitted at *Data Mining and Knowledge Discovery*, revision requested
2. Potolea, R., Trif, F., **Lemnaru, C**., "Adaptive E-Learning Systems with Concept Maps", *Revista Romana de Informatica si Automatica*, vol. 21, no.4/2011, pp. 43-56, 2011
3. Potolea, R., Trif, F., **Lemnaru, C**., "Enhancements on Adaptive E-learning Systems", *The Automation, Computers, Applied Mathematics Journal (ACAM)*, Vol. 19, no.3, pp. 475-482, 2010
4. Potolea, R., **Vidrighin B. C**., Trif, F., "Intelligent Component for adaptive E-learning Systems", *The Automation, Computers, Applied Mathematics Journal*, Vol. 18, pp. 270-275, 2009
5. **C. Vidrighin Bratu** and R. Potolea, "ProICET: a cost-sensitive system for prostate cancer data", *Health Informatics Journal*, Dec 2008, vol. 14: pp. 297-307, ISSN: 1741-2811 (online); 1460-4582

Book chapters

6. **Lemnaru C**. and Potolea R., "Imbalanced Classification Problems: Systematic Study, Issues and Best Practices", to appear in *Lecture Notes in Business Information Processing*, 2012
7. Potolea, R., Barbantan, I., **Lemnaru, C**.,"A Hierarchical Approach for the Offline Handwritten Signature Recognition", *Lecture Notes in Business Information Processing* , 2011, Volume 73, Part 3, 264-279, DOI: 10.1007/978-3-642-19802-1
8. Potolea, R., Cacoveanu, S., **Lemnaru, C.,** "Meta-learning Framework for Prediction Strategy Evaluation", Lecture Notes in Business Information Processing , 2011, Volume 73, Part 3, 280-295, DOI: 10.1007/978-3-642-19802-1
9. **C. Vidrighin Bratu** and R. Potolea, Advances in Greedy Algorithms – chapter 9: "Enhancing Greedy Policy Techniques for Complex Cost-Sensitive Problems", IN-TECH, 2008, pp. 151-168, ISBN 978-953-7619-27-5 (print)

Conference Papers

2012

10. **Lemnaru, C.**, Cuibus, M., Alic, A., Bona, A. and Potolea, R., "A Distributed Methodology for Imbalanced Classification Problems", presented at the *11th International Symposium on Parallel and Distributed Computing,* Munich, June 2012
11. **Lemnaru, C.,** Sin-Neamtiu, A., Veres, M.A., Veres, M and Potolea, R., "A System for Historical Documents Transcription based on Hierarchical Classification and Dictionary Matching", accepted at *KDIR 2012*.
12. **Lemnaru, C.,** Tudose-Vintila, A., Coclici, A. and Potolea, R., "A Hybrid Solution for Imbalanced Classification Problems. Case Study on Network Intrusion Detection", accepted at *KDIR 2012*
13. **Lemnaru, C.,** Dobrin, M., Florea, M., Potolea, R., "Designing a Travel Recommendation System using Case-Based Reasoning and Domain Ontology", accepted *at ICCP 2012.*

2011

14. Potolea, R., **Lemnaru C.**, "A Comprehensive Study of the Effect of Class Imbalance on the Performance of Classifiers" *Proceedings of the 13th International Conference on Enterprise Information Systems*, pp. 14-21, 2011
15. **Lemnaru, C.**, Firte, A., Potolea, R., "Static and Dynamic User Type Identification in Adaptive E-learning with Unsupervised Methods", *Proceedings of the 2011 IEEE 7th International Conference on Intelligent Computer Communication and Processing*, pp. 11-18, 2011
16. Dodut, A.A**., Lemnaru C**. and Potolea R., "The parallel classification of very large collections of data on multi-core platforms", in *Proceedings of the 10th International Symposium on Parallel and Distributed Computing 2011*, pp.57-62, 2011
17. Feier, M., **Lemnaru C.** and Potolea R., "Solving NP-Complete Problems on the CUDA Architecture using Genetic Algorithms", in *Proceedings of the 10th International Symposium on Parallel and Distributed Computing 2011,* pp.278-281, 2011

2010

18. Balla-Muller, N., **Lemnaru, C.,** Potolea, R., "Semi-supervised learning with lexical knowledge for opinion mining," *Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing*, pp.19-25, 2010
19. Bărbănţan, I., **Lemnaru, C.,** Potolea, R., "A Hierarchical Handwritten Offline Signature Recognition System", *Proceedings of the 12th International Conference on Enterprise Information Systems*, Funchal, Madeira, Portugal, pp. 139-147, 2010
20. Cacoveanu, S., **Vidrighin, B. C.,** Potolea, R., "Evaluating Prediction Strategies in an Enhanced Meta-learning Framework", *Proceedings of the 12th International Conference on Enterprise Information Systems*, Funchal, Madeira, Portugal, pp. 148-156, 2010
21. Firte, L., **Lemnaru, C.,** Potolea, R., "Spam detection filter using KNN algorithm and resampling", *Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing*, pp.27-33, 2010
22. Halalai, R., **Lemnaru, C.**, Potolea, R., "Distributed community detection in social networks with genetic algorithms", *Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing*, pp.35-41, 2010
23. R.F. Muresan, **C. Lemnaru**, R. Potolea, "Evidence Combination for Baseline Accuracy Determination", *Proceedings of the 6th ICCP*, pp. 11-18, 2010

24. Potolea, R., **Lemnaru, C**., "Dealing with Imbalanced Problems: Issues and Best Practices", *Proceedings of the 12th International Conference on Enterprise Information Systems*, Volume 2, AIDSS, June 8 - 12, pp. 443-446, ISBN 978-989-8425-05-8, 2010
25. Trif, F., **Lemnaru, C.,** Potolea, R., "Identifying the User Typology for Adaptive E-learning Systems", *Proceedings of AQTR 2010 - IEEE International Conference on Automation, Quality and Testing, Robotics*, Cluj-Napoca, May 28-30, 2010, pp. 192-198/Tome III

2009

26. Bărbănţan, I., **Vidrighin, C.**, Borca, R., "An Offline System for Handwritten Signature Recognition", *Proceedings of the 5th IEEE International Conference on Intelligent computer Communication and Processing*, Cluj-Napoca, pp. 3-10, 2009
27. Cacoveanu, S., **Vidrighin, B.C.,** Potolea, R.,"Evolutional meta-learning framework for automatic classifier selection", *Proceedings of the 5th IEEE International Conference on Intelligent computer Communication and Processing*, pp. 27-30, 2009
28. Firte, A.A., **Vidrighin**, **B.C**., Cenan, C., "Intelligent component for adaptive E-learning systems", *Proceedings of the 5th IEEE International Conference on Intelligent computer Communication and Processing*, pp. 35-38, 2009
29. Merk, A.B., **Vidrighin**, **B.C.,** Potolea, R., "Meta-learning enhancements by data partitioning", *Proceedings of the 5th IEEE International Conference on Intelligent computer Communication and Processing*, pp. 59-62, 2009
30. **Vidrighin, B.C.,** Potolea, R.,"Towards a Unified Strategy for the Preprocessing Step in Data Mining", *Proceedings of the 11th International Conference on Enterprise Information Systems*, pp. 230-235, 2009
31. **Vidrighin, B.C.,** Potolea, R., "Unified Strategy for Feature Selection and Data Imputation", *Proceedings of the 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, Timisoara, pp. 413 – 419, 2009

2008

32. **Vidrighin Bratu, C**., Muresan, T. and Potolea, R., "Improving Classification Performance on Real Data through Imputation", *Proceedings of the 2008 IEEE International Conference on Automation, Quality and Testing, Robotics*, 22-25 May, Cluj-Napoca, Romania, Vol. 3, pp. 464-469, ISBN: 978-1-4244-2576-1
33. **Vidrighin, B. C**, Potolea, R., "Towards a Combined Approach to Feature Selection", *Proceedings of the 3rd International Conference on Software and Data Technologies*, pp. 134-139, 2008
34. **Vidrighin**,B.C., Muresan,T., Potolea,R.,"Improving Classification Accuracy through Feature Selection", *Proceedings of the 4th IEEE International Conference on Intelligent computer Communication and Processing*, pp 25-32

2007

35. T. Moldovan, **C. Vidrighin**, I. Giurgiu, R. Potolea, "Evidence Combination for Baseline Accuracy Determination", *Proceedings of the 3rd IEEE International Conference on Intelligent computer Communication and Processing*, pp. 41-48, 2007
36. **C. Vidrighin Bratu**, C. Savin, R. Potolea, "A Hybrid Algorithm for Medical Diagnosis". *Proceedings of EUROCON 2007*, 9-12 September, Warsaw, pp. 668-673
37. A. Onaci, C. Vidrighin, M Cuibus and R. Potolea, "Enhancing Classifiers through Neural Network Ensembles", *Proceedings of the 3rd IEEE International Conference on Intelligent computer Communication and Processing*, pp. 57-64, 2007

38. R. Potolea, **C. Vidrighin**, C. Savin, "ProICET - A Cost-Sensitive System for the Medical Domain", *Proceedings of the 3rd International Conference on Natural Computation ICNC 2007*, Haikou, August 2007, China, Volume 2, Session 3
39. **C. Vidrighin**, R. Potolea, I. Giurgiu, M. Cuibus, "ProICET: Case Study on Prostate Cancer Data", *Proceedings of the 12th International Symposium of Health Information Management Research*, 18-20 July 2007, Sheffield, pp. 237-244

Citations

C. Vidrighin Bratu, C. Savin, R. Potolea, "A Hybrid Algorithm for Medical Diagnosis". Proceedings of EUROCON 2007, 9-12 September, Warsaw, pp. 668-673 (2007)

- Barros, R. C., Basgalupp, M. P., de Carvalho, A. C. P. L. F., Freitas, A. A., "A Survey of Evolutionary Algorithms for Decision-Tree Induction", *in IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. PP, issue 99, pp. 1-10 (I.F. according to UEFISCDI = 1.6148)
- Weiss, Y., Elovici, Y., Rokach, L., "The CASH algorithm-cost-sensitive attribute selection using histograms", *Information Sciences*, Available online 2 February 2011, ISSN 0020-0255, 10.1016/j.ins.2011.01.035 (I.F. according to UEFISCDI = 1.2769) http://www.sciencedirect.com/science/article/pii/S0020025511000624

C. Vidrighin Bratu, T. Muresan and R. Potolea, "Improving Classification Performance on Real Data through Imputation", *Proceedings of the 2008 IEEE International Conference on Automation, Quality and Testing, Robotics*, 22-25 May, Cluj-Napoca, Romania, Vol. 3, pp. 464-469 (2008)

- Gheyas, I.A., Smith, L.S., "A neural network-based framework for the reconstruction of incomplete datasets", Neurocomputing, Volume 73, Issues 16–18, October 2010, Pages 3039-3065, ISSN 0925-2312, 10.1016/j.neucom.2010.06.021, (I.F. according to UEFISCDI = 0.86616) http://www.sciencedirect.com/science/article/pii/S0925231210003188
- Gheyas, I.A., *Novel Computationally Intelligent Machine Learning Algorithms for Data Mining and Knowledge Discovery*, PhD Thesis, University of Stirling, 2009 http://hdl.handle.net/1893/2152

C. Vidrighin, R. Potolea, I. Giurgiu, M. Cuibus, "ProICET case study on prostate cancer data", *Proceedings of the 12th International Symposium of Health Information Management Research*, 2007, pp. 237–244. (2007)

- Weiss, Y., Elovici, Y., Rokach, L., "The CASH algorithm-cost-sensitive attribute selection using histograms", *Information Sciences*, Available online 2 February 2011, ISSN 0020-0255, 10.1016/j.ins.2011.01.035 (I.F. according to UEFISCDI = 1.2769) http://www.sciencedirect.com/science/article/pii/S0020025511000624

C. Vidrighin Bratu, T. Muresan and R. Potolea, "Improving Classification Accuracy through Feature Selection", Proceedings of the 4th IEEE International Conference on Intelligent Computer Communication and Processing, ICCP 2008, 28-30 August, pp. 25-32 (2008)

- Marcano-Cedeño, A., Quintanilla-Domínguez, J., Cortina-Januchs, M.G., Andina, D.," Feature selection using Sequential Forward Selection and classification applying Artificial Metaplasticity Neural Network" , *Proceedings of IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society,* 2845 - 2850

- Špečkauskienė, V. and Lukoševičius, A, “A data mining methodology with preprocessing steps”, *Information Technology and Control*, Vol.38, No.4, 2009, 319-324.

S. Cacoveanu, **C. Vidrighin** and R. Potolea, "Evolutional meta-learning framework for automatic classifier selection", Proceedings of the IEEE 5th International Conference on Intelligent Computer Communication and Processing, ICCP2009, 27-29 August, pp. 27-30 (2009)

- Hilario,M., Nguyen, P., Do, H., Woznica, A. and Kalousis, A., “Ontology-Based Meta-Mining of Knowledge Discovery Workflows”, *Meta-Learning in Computational Intelligence*, series Studies in Computational Intelligence, vol. 358, Springer, 2011, 273-315

Appendix A – Description of the Datasets Employed in the Experiments

Datasets from Chapter 3

Table A.3.1 – Datasets used in the evaluations on data imputation

Dataset	No. Attributes	No. Instances	Attributes type
Pima Indian Diabethes	8+1	768	Num[7]
Cars	6+1	1729	Nom[8]

Datasets from Chapter 4

Table A.4.1 – Datasets used in the evaluations on wrapper feature selection (section 4.4.1)

Dataset	No. Attributes	No. Instances	Attributes type
Australian	14+1	690	Num, Nom
Breast-cancer	9+1	286	Nom
Bupa	5+1	345	Num
Cleve-detrano	14+1	303	Num, Nom
Crx	15+1	690	Num, Nom
German	20+1	1000	Num, Nom
Heart	13+1	270	Num, Nom
Cleveland	13+1	303	Num, Nom
Monk3	7+1	432	Nom
Pima Diabethes	8+1	768	Num
Thyroid	20+1	7200	Num, Nom
Tic-tac-toe	9+1	958	Nom
Vote	16+1	435	Nom
Wisconsin	9+1	699	Num

Table A.4.2 – Datasets used in the evaluations on the search combination method for wrapper feature selection (section 4.4.2)

Dataset	No. Attributes	No. Instances	Attributes type
Australian	14+1	690	Num, Nom
Breast-cancer	9+1	286	Nom
Cleve-detrano	14+1	303	Num, Nom
Crx	15+1	690	Num, Nom
German	20+1	1000	Num, Nom
Heart	13+1	270	Num, Nom
Hepatitis	19+1	155	Num, Nom
Labor	16+1	57	Num, Nom
Lymphography	18+1	148	Nom
Pima Diabethes	8+1	768	Num
Tic-tac-toe	9+1	958	Nom

[7] Numeric
[8] Nominal

Datasets from Chapter 5

Table A.5.1 – Datasets used in the evaluations on the combined pre-processing strategy (section 5.2)

Dataset	No. Attributes	No. Instances	Attributes type
Bupa	5+1	345	Num
Cleveland Heart Disease	13+1	303	Num, Nom
Pima Diabethes	8+1	768	Num

Datasets from Chapter 6

Table A.6.1 – Datasets employed in the experiments on ProICET (section 6.3.3)

Dataset	No. Att.	Att. Domain	No. Inst.	No. Classes
Breast Cancer Wisconsin	10	Nom	699	2
Bupa Liver Disorder	6	Num	345	2
Pima Indian Diabethes	9	Num	768	2
Cleveland Heart Disease	14	Num, Nom	303	5
Thyroid	21	Num, Nom	7200	3

Table A.6.2 – Attribute information and costs for the Heart Disease Cleveland dataset

Crt. No.	Attribute Name	Attribute Description	Attribute Domain	Attribute Cost
1	Age	Patient age	Numeric	1.0
2	Sex	Patient gender	Binary: 0/1	1.0
3	Cp	Chest Pain Type	Nominal: Value 1: typical angina Value 2: atypical angina Value 3: non-anginal pain Value 4: asymptomatic	1.0
4	Trestbps	Resting Blood Pressure	Numeric	1.0
5	Chol	Serum cholestoral	Numeric	7.27
6	Fbs	Fasting Blood Sugar	Binary: 0/1	5.20
7	Restecg	Resting EKG results	Binary: Value 0: normal elevation of depression of >0.05 mV Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria	15.50
8	Thalach	Maximum rate achieved	Numeric	102.90
9	Exang	Exercise induced angina	Binary: 0=no; 1=yes	87.30
10	Oldpeak	ST depression induced by exercise relative to rest	Numeric	87.30

Crt. No.	Attribute Name	Attribute Description	Attribute Domain	Attribute Cost
11	Slope	The slope of the peak exercise ST segment	Nominal: Value 1: up-sloping Value 2: flat Value 3: down-sloping	87.30
12	Ca	Number of major vessels	Nominal: 0-3	100.90
13	Thal		Nominal: Value 3: normal Value 6: fixed effect Value 7: reversible effect	102.90
14	Class	Diagnosis of heart disease	Nominal: Value 0: missing Values 1-4: degrees of seriousness of heart disease.	0.0

Cost matrix:

$$Cost\_matrix = \begin{array}{c} \begin{matrix} 0 & 1 & 2 & 3 & 4 \end{matrix} \leftarrow \text{classified as} \\ \begin{pmatrix} 0.0 & 10.0 & 20.0 & 30.0 & 40.0 \\ 50.0 & 0.0 & 10.0 & 20.0 & 30.0 \\ 100.0 & 50.0 & 0.0 & 10.0 & 20.0 \\ 150.0 & 100.0 & 50.0 & 0.0 & 10.0 \\ 250.0 & 150.0 & 100.0 & 50.0 & 0.0 \end{pmatrix} \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{matrix} \\ \uparrow \textit{actual class} \end{array}$$

Table A.6.3 – Attribute information and costs for the Bupa Liver Disorder dataset

Crt.No.	Attribute Name	Attribute Description	Attribute Domain	Attribute Cost
1	Mcv	Mean corpuscular volume	Numeric	7.27
2	Alkphos	Alkaline Phosphotase	Numeric	7.27
3	Sgpt	Alamine Aminotransferase	Numeric	7.27
4	Sgot	Aspartate Aminotransferase	Numeric	7.27
5	Gammagt	Gamma-glutamyl Transpeptidase	Numeric	9.86
6	Drinks	Number of half-pint equivalents of alcoholic beverages drunk per day (class attribute)	Binary: ‘less than 3’, ‘more than 3’	0.0
7	Selector	Not used	-	-

Cost matrix:

$$Cost\_matrix = \begin{array}{c} \begin{matrix} \leq 3 & > 3 \end{matrix} \leftarrow \textit{classified as} \\ \begin{pmatrix} 0.0 & 5.0 \\ 15.0 & 0.0 \end{pmatrix} \\ \uparrow \textit{actual class} \end{array}$$

Table A.6.4– Attribute information and costs for the Thyroid dataset

Crt. No.	Attribute Name	Attribute Description	Attribute Domain	Attrib. Cost
1	Age	Age in years	Numeric	1.00
2	Sex	Gender	Binary: 0/1	1.00
3	On_Thyroxin	Patient on thyroxin	Binary: 0/1	1.00
4	Query_On_Thyroxin	Query on thyroxin	Binary: 0/1	1.00
5	On_AntiThyroid_Med	On antithyroid medication	Binary: 0/1	1.00
6	Sick	Patient reports malaise	Binary: 0/1	1.00
7	Pregnant	Patient is pregnant	Binary: 0/1	1.00
8	Thyroid_Surgery	Thyroid surgery history	Binary: 0/1	1.00
9	I131_treatment	On I131 treatment	Binary: 0/1	1.00
10	Query_Hypothyroid	Maybe Hypothyroid	Binary: 0/1	1.00
11	Query_Hyperthyroid	Maybe Hyperthyroid	Binary: 0/1	1.00
12	Lithium	Patient on lithium	Binary: 0/1	1.00
13	Goitre	Patient has goitre	Binary: 0/1	1.00
14	Tumor	Patient has tumor	Binary: 0/1	1.00
15	Hypopituitary	Patient hypopituitary	Binary: 0/1	1.00
16	Psych	Psychological Symptoms	Binary: 0/1	1.00
17	TSH	TSH value	Numeric	22.78
18	T3	T3 value	Numeric	11.41
19	TT4	TT4 value	Numeric	14:51
20	T4U	T4U value	Numeric	11:41
21	FTI	FTI – computed from20 and 21	Not used	-
22	Class	Diagnostic class	Nominal: Value 3: not ill Value 2: hyperthyroid Value 1: hypothyroid	0.0

Cost matrix:

$$Cost\_matrix = \begin{array}{c} \begin{matrix} 3 & 2 & 1 \end{matrix} \leftarrow classified\ as \\ \begin{pmatrix} 0.0 & 5.0 & 7.0 \\ 12.0 & 0.0 & 5.0 \\ 20.0 & 12.0 & 0.0 \end{pmatrix} \begin{matrix} 3 \\ 2 \\ 1 \end{matrix} \\ \uparrow\ actual\ class \end{array}$$

Table A.6.5 – Attribute information and costs for the Pima Indian Diabethes dataset

Crt. No.	Attribute Name	Description	Domain	Cost
1	NbTimesPreg	Number of times pregnant	Numeric	1.0
2	GlucTest	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	Numeric	17.61
3	BloodPress	Diastolic blood pressure (mm Hg)	Numeric	1.0
4	SkinThickness	Triceps skin fold thickness (mm)	Numeric	1.0
5	SerumInsuline	2-Hour serum insulin (mu U/ml)	Numeric	22.78
6	BodyMassIndex	Body mass index (weight in kg/(height in m)^2)	Numeric	1.0
7	PedigreeFct	Diabetes pedigree function	Numeric	1.0
8	Age	Patient age (years)	Numeric	1.0
9	Class	Class attribute	Binary: 0-absence, 1-presence	0.0

Cost matrix:

$$Cost\_matrix = \begin{array}{c} \begin{matrix} 0 & 1 \end{matrix} \leftarrow \textit{classified as} \\ \begin{pmatrix} 0.0 & 7.0 \\ 20.0 & 0.0 \end{pmatrix} \begin{matrix} 0 \\ 1 \end{matrix} \\ \uparrow \textit{actual class} \end{array}$$

Table A.6.6 – Attribute information and costs for the Prostate Cancer dataset

Crt. No.	Attribute Name and Description	Domain
1	One (*TNM*)	Symbolic (1a, 1b, 1c, 2a, 2b, 3a, 3b)
2	Two (*Gleason Score*)	Numeric (2-10)
3	Three (*Presence on Median Intravesical Lobe*)	Symbolic (not present in the ultrasound, voluminous, intravesical)
4	Four (*Prostate Volume*)	Numeric
5	Five (*Preoperative PSA*)	Numeric (ng/ml)
6	Six (*IIEF - International Index of Erectile Function*)	Numeric
7	Seven (*Quality of Life*)	Numeric (0-2)
8	Eight (*Surgery Type*)	Symbolic (TP, EP)
9	Nine (*Operative Technique*)	Symbolic (Ante Grade, Retro Grade, Bipolar)
10	Ten (*Nerve Sparing*)	Symbolic (Non, NS Left, NS Right)
11	Eleven (*Bleeding*)	Numeric (minutes)
12	Twelve (*Anastomosis*)	Symbolic (Continuous, Separate, Van Velt)
13	Thirteen (*Operative Time*)	Numeric (minutes)
14	Fourteen (*Postoperative Hospitalization*)	Numeric (days)
15	Fifteen (*Complications*)	Boolean
16	*Class (Postoperative PSA)*	Symbolic (low: PSA < 0.1, medium: PSA $\in$ [0.1, 1], high: PSA > 1)

Cost matrices

$$Cost\_matrix\,1 = \begin{array}{c} \begin{matrix} low & med & high \end{matrix} \leftarrow classified\ as \\ \begin{pmatrix} 0.0 & 0.5 & 1.0 \\ 1.5 & 0.0 & 0.7 \\ 5.0 & 3.0 & 0.0 \end{pmatrix} \begin{matrix} low \\ med \\ high \end{matrix} \\ \uparrow actual\ class \end{array}$$

$$Cost\_matrix\,2 = \begin{array}{c} \begin{matrix} low & med & high \end{matrix} \leftarrow classified\ as \\ \begin{pmatrix} 0.0 & 0.5 & 1.0 \\ 3.0 & 0.0 & 0.7 \\ 10.0 & 6.0 & 0.0 \end{pmatrix} \begin{matrix} low \\ med \\ high \end{matrix} \\ \uparrow actual\ class \end{array}$$

$$Cost\_matrix\,3 = \begin{array}{c} \begin{matrix} low & med & high \end{matrix} \leftarrow classified\ as \\ \begin{pmatrix} 0.0 & 0.5 & 1.0 \\ 0.75 & 0.0 & 0.7 \\ 2.5 & 1.5 & 0.0 \end{pmatrix} \begin{matrix} low \\ med \\ high \end{matrix} \\ \uparrow actual\ class \end{array}$$

$$Cost\_matrix\,4 = \begin{array}{c} \begin{matrix} low & med & high \end{matrix} \leftarrow classified\ as \\ \begin{pmatrix} 0.0 & 0.5 & 1.0 \\ 3.0 & 0.0 & 0.5 \\ 5.0 & 3.0 & 0.0 \end{pmatrix} \begin{matrix} low \\ med \\ high \end{matrix} \\ \uparrow actual\ class \end{array}$$

Number of instances: 389
Missing values: yes

Datasets from Chapter 7

Table A.7.1: Benchmark datasets employed in the experiments on the effect of class-imbalance on the performance of classifiers (section 7.1.3)

Dataset	No. Att.	No. Inst.	IR	IAR	C		Dataset	No. Att.	No. Inst.	IR	IAR	C
Bupa	6	345	1	58	3		*Ecoli_im_rm*	8	336	3	42	2
Haberman_1	4	367	1	92	3		*Glass_NW*	11	214	3	19	4
Cleve	14	303	1	22	5		*Vehicle_van*	19	846	3	45	4
Monk3	7	554	1	79	4		*Chess_IR5*	37	2002	5	54	5
Monk1	7	556	1	79	5		*Segment_1*	20	1500	6	75	3
Australian	15	690	1	46	5		*Ecoli_imu*	8	336	9	42	4
Crx	16	690	1	43	5		*Segment_1 _IR10*	20	1424	10	71	3
Chess	37	3196	1	86	5		*Tic-tac-toe_IR10*	10	689	10	69	6
Mushrooms	23	8124	1	353	4		*German_IR10*	21	769	10	37	7
Breast-cancer	10	286	2	29	2		*Sick-euthyroid*	26	3163	10	122	5
Glass_BWNFP	11	214	2	19	3		*Glass_VWFP*	11	214	12	19	3
Glass_BWFP	11	214	2	19	4		*Sick*	30	3772	15	126	5
Vote	17	435	2	26	3		*Ecoli_bin*	8	336	16	42	3
Wisconsin	10	699	2	70	4		*Caravan*	86	5822	16	68	11
Pima	7	768	2	110	4		*Ecoli_im_rm*	8	336	3	42	2
Tic-tac-toe	10	958	2	96	7		*Glass_NW*	11	214	3	19	4
German	21	1000	2	48	7		*Vehicle_van*	19	846	3	45	4

A number of multi-class problems were modified to obtain binary classification problems from multi-class data. Also, three of the relatively large datasets were under-sampled to generate higher IR values (contain *_IR* in their name). The complexity of each dataset was approximated, as suggested in (Jap02), to $C = \log_2 L$, where L is the number of leaves generated by the C4.5 decision tree learner. Also, the values for IR, IAR and C have been rounded.

Table A.7.2 – Large IR, small IAR datasets, employed in the experiments on ECSB (section 7.3.2)

Dataset	#Examples	#Attributes	IR	IAR
Chess_IR5	2002	37	5	54
Ecoli_om_remainder_binary	336	8	15.8	42
Ecoli_imu_remainder_binary	336	8	8.6	42
Glass_VWFP_binary	214	10	11.59	21
German_IR10	769	21	10.14	37

Table A.7.3 – Datasets employed for comparison of ECSB with EUS (section 7.3.2)

Dataset	#Examples	#Attrs	IR	Dataset	#Examples	#Attrs	IR
GlassBWNFP	214	9	1.82	*Optdigits0*	5,564	64	9.1
EcoliCP-IM	220	7	1.86	*Satimage4*	6,435	36	9.28
Pima	768	8	1.9	*Vowel0*	990	13	10.1
GlassBWFP	214	9	2.06	*GlassVWFP*	214	9	10.39
German	1,000	20	2.33	*EcoliOM*	336	7	13.84
Haberman	306	3	2.68	*GlassContainers*	214	9	15.47
Splice-ie	3,176	60	3.15	*Abalone9-18*	731	9	16.68
Splice-ei	3,176	60	3.17	*GlassTableware*	214	9	22.81
GlassNW	214	9	3.19	*YeastCYT-POX*	483	8	23.15
VehicleVAN	846	18	3.25	*YeastME2*	1,484	8	28.41
EcoliIM	336	7	3.36	*YeastME1*	1,484	8	32.78
New-thyroid	215	5	4.92	*YeastEXC*	1,484	8	39.16
Segment1	2,310	19	6	*Car*	1,728	6	71.94
EcoliIMU	336	7	8.19	*Abalone19*	4,177	9	128.9

Table A.7.4 – Datasets employed for comparison of ECSB with the SVM ensemble (section 7.3.2)

Dataset	#Examples	# Attributes	IR
Breast-cancer	286	10	2.36
Cars	1729	7	25.2
Glass-headlamps	214	9	6.38
Balance-scale	625	5	11.76

Appendix B – Relevant Research Papers

The following pages provide a listing of three relevant research papers for the work presented in the current thesis.