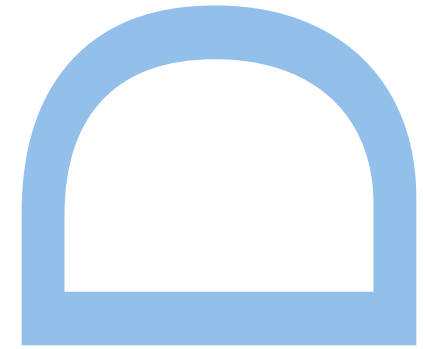
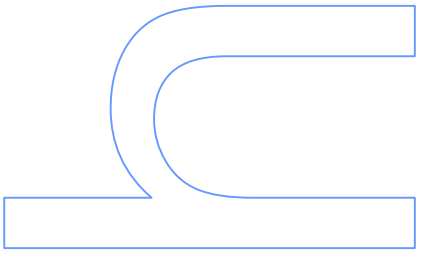
Utility-based Predictive Analytics

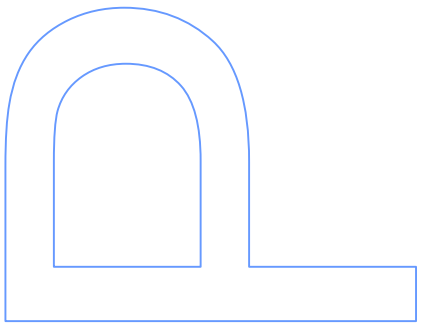
Paula Alexandra de Oliveira Branco
Programa Doutoral em Informática das
Universidades do Minho, Aveiro e Porto
Departamento de Ciência de Computadores
2018



Orientador
Luís Fernando Rainho Alves Torgo, Professor Associado
Faculdade de Ciências da Universidade do Porto

Coorientador
Rita Paula Almeida Ribeiro, Professora Auxiliar
Faculdade de Ciências da Universidade do Porto





Author's declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: .....

DATE: .....

To Nicolau and Leonardo

Acknowledgments

First and foremost, I would like to thank my supervisors, Professor Luís Torgo and Professor Rita Ribeiro, for all the support, guidance and encouragement they have provided me since I started this life-changing journey. They have made possible the realisation of this thesis. It has been an honour to work with both. I thank Professor Luís Torgo for always expressing his thoughts, concerns and suggestions in a clear and direct way. Professor Rita Ribeiro was always patient, available and willing to help in the most difficult moments. I learned a lot with both and I am thankful for the excellent examples they provided as successful researchers and professors. For all this, they have my sincere respect and gratitude.

My thanks also go to my colleagues in DCC and LIAAD - INESC Tec for the many wonderful moments but also for providing me support on the most difficult times. A special word to Mariana Oliveira, Nuno Moniz and Vitor Cerqueira.

Lastly, I would like to thank my family for all the love and support. A special thanks to my mother Amélia, my husband Nicolau and my son Leonardo. A very special thanks to my loving, encouraging, and tremendously patient husband Nicolau. Without his confidence and faithful support this journey would not have been possible. I am also very grateful to my son Leonardo for all the happiness he has brought into my life.

The work presented in this thesis was supported by a scholarship from the Portuguese Science and Technology Foundation (FCT) with the PhD grant PD/BD/105788/2014, by the ERDF – European Regional Development Fund through the COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, and by National Funds through FCT as part of project UID/EEA/50014/2013.

Abstract

In several predictive tasks the end-user attention is focused in certain regions of the domain of the target variable. As opposed to standard predictive tasks where all target variable values are equally important, in these particular tasks the domain has a non-uniform importance for the end-user. In many real world domains, such as financial, meteorological or medical, we can find tasks that fit into this non-standard setting. In effect, for most practical applications we observe that there is important domain knowledge that must be accounted for when solving the corresponding predictive task.

In these tasks, the relevance of certain regions of the domain is associated to either high costs and/or severe consequences, or to important profits and/or benefits. Initially, the research community addressed these tasks through the development of the cost-sensitive learning theory which considers only the costs component. More recently, this theory evolved to the broader framework of utility-based mining. The utility-based learning setting allows the consideration of both costs and benefits that may derive from different domain information. Although more complex, the utility-based learning framework is also more intuitive from an end-user perspective and more thorough regarding the domain information representation.

The first efforts for including costs and benefits into the learning procedure were concentrated in tasks with a nominal target variable (classification tasks). Still, with time, it became clear that this learning paradigm was also applicable to regression tasks, where the target variable is continuous. In this thesis we focus on the utility-based learning problem. The youth of this broader approach, specially regarding regression tasks, results in the existence of several open issues that we tackled. In particular, we identified and addressed the following main challenges: i) development of a unifying framework for utility-based learning; ii) development of learning methods for optimising utility in regression tasks; and iii) proposal of new pre-processing methods for addressing the problem of learning from imbalanced domains.

The proposal of a unifying utility-based framework allows to better understand the characteristics of these tasks, while integrating both classification and regression problems. Moreover, using this framework we are also able to establish important connections between different predictive problems.

The second challenge is related with the lack of methods to address utility-based regression

problems. When dealing with utility-based learning if the utility information is not incorporated into the learning procedure we are only able to obtain sub-optimal models. To solve this problem we propose and evaluate new methods that allow to maximise the utility in regression. We show that these methods are effective for different utility settings.

The third and final challenge is related with the particular sub-class of utility-based problems known as imbalanced domains. This is also a problem insufficiently studied in the context of regression tasks. We propose and evaluate several approaches to address this problem.

As a practical outcome of this work, we provide UBL, an R package for utility-based learning that integrates approaches for classification and regression tasks. The UBL package includes the proposals presented in this thesis, as well as many other approaches developed for utility-based classification, providing to the research community a tool for testing and comparing different alternative methods of addressing utility-based predictive tasks.

Resumo

Em várias tarefas de previsão a atenção do utilizador final está centrada em certas regiões do domínio da variável objetivo. Ao contrário do que sucede com as tarefas de previsão *standard* nas quais todos os valores a variável objetivo são igualmente importantes, nestas tarefas o domínio tem uma importância não uniforme para o utilizador final. Em muitos domínios reais, como o financeiro, meteorológico ou médico, podemos encontrar tarefas que se enquadram nesta configuração não *standard*. Com efeito, observamos que na maioria das aplicações práticas existe conhecimento sobre o domínio importante que deve ser tido em conta no momento da resolução da tarefa de previsão correspondente.

Nestas tarefas, a relevância de certas regiões do domínio está associada a elevados custos e/ou graves consequências, ou a lucros e/ou benefícios consideráveis. Inicialmente, a comunidade científica abordou estas tarefas através do desenvolvimento da teoria de *cost-sensitive learning* a qual considera apenas a componente dos custos. Recentemente, esta teoria evoluiu para a estrutura mais ampla de *utility-based mining*. Esta configuração de aprendizagem baseada em utilidade permite considerar simultaneamente custos e benefícios que podem derivar de diferentes informações sobre o domínio. Apesar de ser mais complexa, a estrutura de aprendizagem baseada em utilidade é também mais intuitiva do ponto de vista do utilizador final e mais completa em relação à representação de informação sobre o domínio.

Os primeiros esforços para incluir custos e benefícios no processo de aprendizagem estavam concentrados em tarefas com uma variável objetivo nominal (tarefas de classificação). No entanto, com o tempo, tornou-se claro que este paradigma de aprendizagem era também aplicável a tarefas de regressão, as quais têm a variável objetivo contínua. Nesta tese focámo-nos no problema de aprendizagem baseada em utilidade. A juventude desta ampla abordagem, especialmente no que diz respeito a tarefas de regressão, resulta na existência de diversas questões em aberto que nos enfrentámos. Em particular, identificámos e abordámos os principais desafios seguintes: i) desenvolvimento de uma estrutura unificadora para aprendizagem baseada em utilidade; ii) desenvolvimento de métodos de aprendizagem para otimizar a utilidade em tarefas de regressão; e iii) proposta de novos métodos de pré-processamento para lidar com o problema de aprendizagem em domínios desbalanceados.

A proposta de uma estrutura unificadora para aprendizagem baseada em utilidade permite compreender melhor as características destas tarefas, integrando ambos os problemas de classificação e regressão. Além disso, usando esta estrutura, também podemos estabelecer conexões importantes entre diferentes problemas de previsão.

O segundo desafio está relacionado com a falta de métodos para abordar problemas de regressão baseados em utilidade. Ao lidar com problemas de aprendizagem baseados em utilidade, se a informação relativa à utilidade não é incorporada no processo de aprendizagem seremos apenas capazes de obter modelos sub-ótimos. Para resolver este problema propomos e avaliamos métodos novos que permitem maximizar a utilidade em regressão. Mostramos que estes métodos são eficazes em diferentes configurações de utilidade.

O terceiro e último desafio está relacionado com uma subclasse particular de problemas baseados em utilidade conhecida por domínios desbalanceados. Este é também um problema estudado de forma insuficiente no contexto de tarefas de regressão. Nós propomos e avaliamos abordagens para lidar com este problema.

Como resultado prático deste trabalho, disponibilizamos a UBL, uma package de R para aprendizagem baseada em utilidade que integra abordagens para tarefas de classificação e regressão. A package UBL inclui as abordagens apresentadas nesta tese, assim como muitas outras abordagens desenvolvidas para tarefas de classificação baseadas em utilidade, fornecendo à comunidade científica uma ferramenta para testar e comparar diferentes métodos alternativos para abordar tarefas de previsão baseadas em utilidade.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

The growth witnessed in the amount of available data has been accompanied by the will of using it in a valuable way. In a Data Science context, this growth also brought attention to the importance of taking into account all existing domain knowledge. The incorporation of domain information when dealing with predictive tasks is crucial for many practical applications, leading to more adequate and useful predictions.

Frequently, the end-user is particularly interested in some specific cases. This information regarding important domain regions, represents situations where high profits or benefits may occur, but also serious consequences or costs may be at stake. In this scenario, the goal is to incorporate the available domain knowledge in the generation of the models. These models should be able to make predictions that provide the higher utility value for the end-user taking into account her/his domain preference biases.

This thesis addresses the problem of utility-based predictive analytics. The goal here is to predict a target variable value using its dependency on a set of predictor variables, while incorporating the available domain knowledge to maximise a utility score.

In this chapter we contextualise and define the problem of utility-based predictive analytics. Our main motivations and contributions are also outlined.

1.1 Data Science and Predictive Analytics

The increasing amount of available data has promoted the fast development of Data Science. The key objective of Data Science is to extract potentially interesting and novel knowledge from the available data. It is this capability of Data Science to discover knowledge that makes it so attractive in a diversity of practical domains. In effect, the discovered knowledge may provide an important competitive advantage for many organisations and different application domains, such as: medical, financial markets, meteorology/ecology, among others.

Data Science encompasses several different tasks among which predictive analytics occupies a leading position. Predictive analytics aims at building models for solving predictive tasks using examples of the domain.

In the course of time, the research community verified that standard learning tasks using only the domain examples as input were not the most common setting. Typically, for a given predictive task there is also other important domain knowledge that is crucial to consider. For instance, it is frequent to find practical applications where the end-user interests are biased towards specific regions of the domain which may provide high profits or costs. This extra information must be accounted for when providing solutions for the predictive tasks. These non-standard predictive tasks are more complex as they involve embedding more domain knowledge into the learning process. However, if this knowledge is disregarded the developed models will not be suitable for the end-user goals. Solving such tasks is an interesting problem that still presents important open research challenges.

1.2 Context and Problem Definition

This thesis is focused on a particular type of Data Science applications: predictive tasks where the main goal is to maximise the utility of the predictions. These tasks include predictive problems for which the end-user is able to provide information on hers/his preference biases in a formal or informal way. This information involving non-uniform preferences over the problem domain affects both the learning strategies and the evaluation procedures that should be applied. A diverse set of practical applications share the goal of maximising the utility of the predictions. This is the case for the medical diagnosis of rare diseases, forecasting extreme weather events, prediction of high/low values of stock market assets, among many others.

The key distinguishing characteristic of these applications is related with the impact of predictions on the most relevant regions of the domain of the target variable. These predictions usually entail high costs and/or severe consequences, or may represent high profits and/or benefits. A well-known framework for addressing some of these problems is cost-sensitive learning (e.g. Elkan [2001]). Still, this framework is focused on classification problems, where the target variable is nominal, and few solutions exist for regression problems, where the target variable is continuous.

In this thesis we focus on the open issues associated with these non standard tasks, which are more noticeable in regression problems. Examples of such open issues include: the definition of a unifying framework for utility-based learning problems, the development of learning methods for utility optimisation specially in a regression context, or the definition of suitable performance assessment measures.

1.3 Motivation and Main Contributions

In this thesis we address a special type of predictive tasks: utility-based predictive analytics. The goal of this type of tasks is to use the utility information regarding the costs and benefits of predictions supplied by the end-user to obtain models that maximise the utility, i.e., models that provide lower costs and higher benefits with their predictions. The existence of this additional domain knowledge is a key factor in these problems. If this knowledge is disregarded only sub-optimal models can be obtained which may have severe consequences. The several open issues still existing in these utility-based applications, particularly in the less explored regression tasks, motivated the work presented in this thesis.

Solutions for the utility-based learning problem have been mostly focused on classification tasks. Thus, the majority of open challenges are related with utility-based regression tasks. One main objective of this thesis is to address these open issues, providing a diverse set of solutions for utility-based regression tasks. Another important objective of this thesis is to develop a unifying framework incorporating both utility-based classification and regression tasks, that allows to: i) understand the connections between standard and non-standard predictive tasks; and ii) show the relationship between utility-based learning and the problem of learning from imbalanced domains.

The work carried out during the thesis led to the following main contributions:

i) present an extensive review of the utility-based learning problem, including the particular sub-problem of learning from imbalanced domains;

ii) propose a general utility-based learning framework that allows to contextualise these tasks within standard and non-standard learning problems;

iii) frame the problem of learning from imbalanced domains as a sub-problem of utility-based learning;

iv) present and discuss the main open challenges of utility-based learning problems;

v) present a solution based on spatial interpolation for addressing the challenge of obtaining the full utility information in regression problems;

vi) present performance assessment measures adjusted to the available problem information;

vii) propose and evaluate two new methods for maximising utility in regression tasks;

viii) propose and evaluate several new pre-processing methods for dealing with the imbalanced regression problem;

ix) development of UBL, an R package for dealing with utility-based learning problems. The UBL package includes several different methods for tackling utility-based learning problems for both classification and regression tasks and is freely available for the research community.

1.4 Organisation of the Thesis

This thesis is organised in six chapters described below.

Introduction In the present chapter we described this thesis context and problem definition. We also present the main motivations and contributions of our work.

Literature Review The second chapter provides an extensive review of previous work on utility-based predictive analytics. We present the main challenges and developments achieved in these problems which also include the problem of learning with imbalanced domains.

Utility-based Learning Framework The third chapter presents a unifying framework for utility-based learning that incorporates classification and regression tasks. Using this framework we are able to show the relationship between utility-based learning and imbalanced learning problems.

Utility Optimisation for Regression Tasks In the fourth chapter we propose two new methods for optimising the utility in regression problems. An experimental evaluation is presented and a discussion regarding the advantages of the two methods is provided.

Learning from Imbalanced Regression Problems In the fifth chapter, we present a diverse set of pre-processing methods to tackle imbalanced regression problem. The presented methods include strategies for biasing the pre-processing strategies according to the examples neighbourhood. The use of these methods is evaluated in an extensive experimental study and a discussion of the results is provided.

Conclusions The sixth and final chapter concludes this thesis. We present a summary of our main achievements and provide insights regarding future research directions.

1.5 Bibliographic Note

This thesis includes work that has been published elsewhere. The following list provides the references to those publications as well as the chapters to which they are related.

- Paula Branco, Luis Torgo, and Rita P Ribeiro. A survey of predictive modelling under imbalanced distributions. *arXiv preprint arXiv:1505.01658*, 2015 **(Chapter 2, 3)**
- Paula Branco, Luís Torgo, and Rita P Ribeiro. A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys (CSUR)*, 49(2):31, 2016b **(Chapter 2, 3)**
- Paula Branco, Luís Torgo, and Rita P Ribeiro. Relevance-based evaluation metrics for multi-class imbalanced domains. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 698–710. Springer, 2017b **(Chapter 3)**
- Paula Branco, Rita P Ribeiro, and Luis Torgo. UBL: an R package for utility-based learning. *arXiv preprint arXiv:1604.08079*, 2016a **(Chapter 3, 4, 5)**
- Paula Branco, Luís Torgo, Rita P Ribeiro, Eibe Frank, Bernhard Pfahringer, and Markus Michael Rau. Learning through utility optimization in regression tasks. In *Data Science and Advanced Analytics (DSAA), 2017 IEEE International Conference on*, pages 30–39. IEEE, 2017d **(Chapter 4)**
- Paula Branco, Luís Torgo, and Rita P Ribeiro. MetaUtil: Meta learning for utility maximization in regression. In *International Conference on Discovery Science (to appear)*. Springer, 2018a **(Chapter 4)**
- Luís Torgo, Paula Branco, Rita P Ribeiro, and Bernhard Pfahringer. Resampling strategies for regression. *Expert Systems*, 32(3):465–476, 2015 **(Chapter 5)**
- Paula Branco, Luís Torgo, and Rita P Ribeiro. SMOGN: a pre-processing approach for imbalanced regression. In *First International Workshop on Learning with Imbalanced Domains: Theory and Applications*, pages 36–50, 2017c **(Chapter 5)**
- Paula Branco, Luís Torgo, and Rita P Ribeiro. Exploring resampling with neighborhood bias on imbalanced regression problems. In *Portuguese Conference on Artificial Intelligence*, pages 513–524. Springer, 2017a **(Chapter 5)**
- Paula Branco, Luís Torgo, and Rita P Ribeiro. Resampling with neighbourhood bias on imbalanced domains. *Expert Systems*, 2018b **(Chapter 5)**

Chapter 2

Literature Review

Standard predictive analytics assumes the user has a uniform interest across the domain of the target variable. However, the study of real world applications shows that this is not always the case. A diversity of practical applications has drawn the attention of the research community to situations where this interest is non-uniform. This happens in many real world applications such as extreme weather forecasting or prediction of rare diseases.

In this chapter we discuss the main challenges when tackling this particular class of predictive problems and present the main existing solutions to address them. We define non-standard predictive analytics tasks and provide an overview of the methods developed for dealing with them. Namely, we cover some of the most common examples of non-standard predictive analytics: the problems of cost-sensitive learning, utility-based learning and learning from imbalanced domains.

2.1 Introduction

Data Science involves a diverse set of tasks, including data cleaning, data pre-processing and transformation, data visualisation, learning of predictive models, development of decision support tools, deployment of recommendation systems, among other. The main goal of Data Science is to discover new interesting knowledge using the available data. The importance of Data Science is precisely related with this process of uncovering knowledge from data which can provide a competitive advantage for many companies and organisations.

Among the different steps involved in Data Science, predictive analytics emerges as a central task. The goal of predictive analytics is to use a set of examples of a given domain to build models that can solve predictive tasks [Torgo, 2016]. The objective of a predictive task is to obtain a good approximation of an unknown function f that maps the values of a set of predictor variables into the values of a target variable. This approximation is obtained using

a training data set containing observations of both the predictor and target variables.

Standard predictive analytics tasks are focused on obtaining high quality predictions assigning the same importance to all cases in the problem domain. In these problems, the performance of the obtained models is assessed by measuring the prediction errors using standard evaluation metrics. These standard metrics are calculated in a uniform, independent and unbiased fashion. This means that the errors are evaluated without taking into account any domain characteristics or preference bias such as the cases where the errors occurred, the type of error, or the user preferences over the domain. Therefore, standard predictive tasks seek a minimisation of the errors, ignoring differences between types of errors, i.e., they assume that all errors have the same cost.

For several real world domains many predictive tasks are non-standard because they involve non-uniform preferences. In fact, for these tasks different errors may have different costs and accurate predictions may also have different benefits. Examples of such tasks appear on medical domains (e.g. prediction of rare diseases), meteorology (e.g catastrophes prediction), financial markets (e.g. forecast of extreme returns in stock markets), ecology (e.g. prediction of water contamination), fraud detection (e.g. prediction of fraudulent credit card transactions), among many others. For instance, when considering the problem of predicting if a person is healthy or has cancer, it is much more serious (costly) to misclassify as healthy a person that actually has cancer than the reverse. In the first type of error, a person life may be at stake, while on the second type of error it only implies conducting further unnecessary exams on a healthy person.

This thesis is focused on this type of non-standard applications of predictive analytics. Our goal is to explore solutions for solving open challenges related with these tasks. Therefore, in this chapter we will start by addressing several examples of non-standard tasks. We describe the cost-sensitive and utility-based learning tasks and present an overview of challenges and existing solutions for addressing these problems. Next, we describe the related problem of learning from imbalanced data sets. We discuss its connections with cost-sensitive and utility-based learning, again describing existing solutions and open challenges.

2.2 Non-standard Predictive Analytics: Tasks and Challenges

Standard predictive tasks aim at obtaining high quality predictions. In the process to achieve this goal all cases are considered equally important. Non-standard predictive tasks are characterised by involving non-uniform preferences.

These non-uniform preferences may be associated to costs and/or benefits (cost-sensitive or utility-based learning [Elkan, 2001]), to the rarity of certain events or values (imbalanced domains learning [Branco et al., 2015]), or may even be associated with more complex settings

involving the decision maker prediction-based actions (e.g. actionable forecasting [Baía, 2015]). All these problems are examples of non-standard predictive analytics. Utility Based Data Mining (UBDM), as proposed by Weiss et al. [2005] and Zadrozny et al. [2006], also fits into the class of "non-standard" predictive analytics. UBDM allows the consideration of both costs and benefits that can arise from different factors.

In this thesis we focus on utility-based learning tasks, and in particular in utility-based regression. Utility-based learning is an extension of cost-sensitive problems that takes into account both costs and benefits of the predictions of the models. The main goal of utility-based learning problems is to obtain models that maximise the utility of their predictions. This utility describes the overall net balance between the costs and benefits of the predictions. To be able to evaluate this utility we need to obtain information on the user preferences and also to adopt suitable evaluation measures.

The problem of learning from imbalanced domains is another example of non-standard predictive tasks. As in utility-based learning this problem also involves non-uniform preferences over the domain of the target variable. However, this problem has an additional difficulty: the cases that are more interesting to the user are poorly represented in the available data. It is the conjunction of these two characteristics that causes the difficulties in imbalanced domains. This means that imbalanced domains are a special case of utility-based learning. Both are based on non-uniform preferences of the end-user. However, the poor representativeness of the most interesting cases on imbalanced domains makes the learning methods disregard them, which causes a performance degradation precisely on these cases. Further details and more precise formalization of both utility-based learning and imbalanced domains will be given in Chapter 3.

When tackling predictive tasks with non-uniform preferences from the end-user, researchers must face a set of important challenges that are closely related with the nature of the problem. These challenges are : (i) how to express the user preferences in a format useful for biasing the learning of the models; (ii) how to evaluate the performance of the learned models; and (iii) how to learn models biased towards the user preferences.

The first challenge is related with the domain knowledge that is available to the analyst. A frequent setting for cost-sensitive classification tasks is to obtain a cost matrix that specifies the different costs associated with each prediction error. Still, many authors claim that obtaining such matrices may be challenging. In imbalanced domains this is also a challenge because typically we only have some informal information available such as "the minority class is the most important one". As we will see, for utility-based regression this expression of the user preferences is even more challenging.

The second challenge is related with performance evaluation. This is important because the use of standard measures can lead to misleading conclusions on the quality of the obtained models. Utility-based learning tasks aim at maximising the utility, while imbalanced domains

aim at maximising the predictive performance on the most important cases. This means that, in both cases, the issue of performance assessment must be adjusted to reflect the user interests. In effect, standard evaluation measures focus on the average behaviour of the models and therefore fail the goal of considering the user preferences.

Finally, the third challenge concerns obtaining models in accordance with the user preferences. Several strategies exist to force the learners to focus on the most interesting cases. These strategies can be categorised into direct and meta-learning methods [Ling and Sheng, 2011b] based on the moment of the modeling workflow where they are applied. Direct methods include solutions that act internally in the learning algorithm. Their goal is to change the original learner to make it focus on the most important and useful cases. These methods are typically effective but tailored to specific domains and therefore hard to use in different contexts. Meta-learning methods act before or after the learning algorithm is applied. They may adjust the training set distribution to correspond to the user preferences, or act on the predictions changing them. The main advantage of meta-learning methods is their independence of the learning algorithm, which makes them generally applicable.

2.3 Utility-based Learning

Utility-based learning is a special type of non-standard predictive task where we take into consideration the fact that end-users assign different costs and benefits to the predictions of the models. Utility-based learning can be seen as an extension of cost-sensitive learning. Cost-sensitive predictive tasks may involve several different types of costs. Turney [2002] proposed a taxonomy for the possible types of costs. Among the proposed types of costs we find: costs of classification errors, costs of data acquisition, active learning costs, costs of intervention, costs of computation and costs of human-computer interaction. Still, it is recognised that the cost of classification errors is, by far, the most studied type and, therefore, has a major role in the context of cost-sensitive learning.

Having differentiated costs of classification errors is useful in different real world applications, such as, prediction of rare diseases, forecasting of unusual stock returns, credit approval, anomaly detection in sensors data, prediction of meteorological catastrophes, etc. Given the different contexts, costs can be represented through different units. Cost may be monetary, but can also represent an illness severity, a waste of time, a computational cost, among others.

The main goal of cost-sensitive algorithms is to minimise the total cost, as opposed to standard learning tasks that assume uniform costs, and aim at minimising the number of errors.

The more general framework of UBDM allows to consider both costs and benefits across all

stages of the data mining process. The term *utility* emerged in the field of economics where it was used to measure the preferences of the users over a set of goods or services. The goal is the maximisation of the expected utility which can be achieved in multiple ways, by improving the utility results in the different stages of the data mining process.

In the following subsections we present the main theory developed for cost-sensitive and utility-based learning, and describe the methods proposed to address these tasks. Namely, we consider the solutions put forward by the research community concerning the 3 main challenges mentioned before: how to express the domain knowledge; how to properly evaluate the models; and how to obtain them according to the user preferences.

2.3.1 Cost-sensitive and Utility-based Learning Theory

The goal of predictive analytics is to obtain a good approximation of an unknown function $Y = f(X_1, X_2, \cdots, X_p)$ that maps the values of a set of p predictor or feature (independent) variables into the target (dependent) variable values. This function f maps the feature space $\mathcal{X}$ into the target variable space $\mathcal{Y}$, i.e., $f : \mathcal{X} \rightarrow \mathcal{Y}$. The approximation $h(X_1, X_2, \cdots, X_p)$ of the function f, also called a model, is obtained using a training set $\mathcal{D} = \{\langle \mathbf{x}_i, y_i \rangle\}_{i=1}^{N}$ and a selected learning algorithm. The cases $\langle \mathbf{x}_i, y_i \rangle$ in $\mathcal{D}$, also named examples, observations or instances, are assumed to be drawn from the same (unknown) distribution. When the target variable is numeric we face a regression task and when it is nominal we have a classification problem.

The model h is obtained by optimising some preference criterion, usually known as the loss function. These functions are used to assess how good the model h is, i.e., how well it is expected to fit a new set of examples drawn from the same distribution.

Definition 2.3.1 (Loss Function) *A loss function $L : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ provides the loss associated with predicting the value of $\hat{y}$ for a true target variable value of y.*

Several loss functions exist. In classification tasks the 0/1 loss (cf. Equation 2.1) is typically used, while for regression tasks the square loss (cf. Equation 2.2) or the absolute loss (cf. Equation 2.3) are the common choices.

$$L_{0/1}(\hat{y}, y) = I(y \neq \hat{y}) \tag{2.1}$$

where I is the indicator function that returns one if its argument is true and zero otherwise.

$$L_2(\hat{y}, y) = (y - \hat{y})^2 \tag{2.2}$$

$$L_1(\hat{y}, y) = |y - \hat{y}| \tag{2.3}$$

Definition 2.3.2 (Expected Loss) *The expected loss of a model h in an instance $\langle \boldsymbol{x}, y \rangle \in \mathcal{X} \times \mathcal{Y}$ draw from a distribution $\mathcal{D}$ is defined as*

$$E_L(h, \langle \mathbf{x}, y \rangle, \mathcal{D}) := \mathbb{E}_{\mathcal{D}}[L(h(\mathbf{x}), y)] \tag{2.4}$$

When we face a standard predictive task, the optimisation criterion used is the minimisation of the expected loss (cf. Equation 2.4). Therefore, the optimal prediction $y^* \in \mathcal{Y}$ for a given instance $\langle \mathbf{x}, y \rangle$ is determined by the model h as,

$$y^* = \underset{h(\mathbf{x}) \in \mathcal{Y}}{\arg\min} E_L(h, \langle \mathbf{x}, y \rangle, \mathcal{D}) \tag{2.5}$$

In this approach all errors are treated the same way, i.e., they all have the same cost. Still, as we have mentioned, for several real world problems it is important to differentiate the costs of the errors. When a cost function, C, describing the penalties incurred when errors are made, is available, the used optimisation criterion must change. In these "non-standard" settings, the predictive task goal changes from expected loss minimisation to the minimisation of the expected cost.

Definition 2.3.3 (Cost Function) *A cost function $C : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ maps each pair of predicted and true values $(\hat{y}, y)$ into a real number.*

The theory of cost-sensitive learning was first developed for classification tasks. Therefore, we will for now focus on cost-sensitive classification problems.

Consider a classification task with K classes, where $\mathcal{Y} = \{y_1, y_2, \cdots, y_K\}$. Typically, when we face a cost-sensitive classification task, the cost function is expressed as a $K \times K$ matrix, that is also called a cost matrix. This formulation (e.g. Breiman et al. [1984], Domingos [1999], Elkan [2001]) allows to express the existing domain knowledge by setting the costs of all possible errors.

Definition 2.3.4 (Cost Matrix) *Consider a classification task with K different classes. A cost matrix $C := [c_{ij}]$ specifies the costs incurred when an example is predicted to be of class i when it actually belongs to class j. This matrix can be more formally defined as follows:*

$$\begin{aligned} C : K \times K &\to \mathbb{R}_0^+ \\ (i, j) &\to c_{ij} = \begin{cases} 0 & i = j \\ > 0 & i \neq j \end{cases} \end{aligned} \tag{2.6}$$

In this setting, accurate predictions should have no cost associated ($c_{ii} = 0$), while misclassified classes should have a positive cost ($c_{ij} > 0$ if $i \neq j$).

Given a classification problem with K classes and a $K \times K$ cost matrix, the goal of the classifier is to minimise the expected cost.

Definition 2.3.5 (Expected Classification Cost) *Given a cost matrix, C, the expected cost of a classifier h in an instance $\langle \boldsymbol{x}, y \rangle \in \mathcal{X} \times \mathcal{Y}$ draw from a distribution $\mathcal{D}$ is defined as*

$$E_{CC}(h, \langle \boldsymbol{x}, y \rangle, \mathcal{D}) := \mathbb{E}_{\mathcal{D}}[C(h(\boldsymbol{x}), y)] \tag{2.7}$$

Consider an instance $\mathbf{x} \in \mathcal{X}$ with unknown class label, for which we want to obtain a class prediction. There are several different ways to achieve this goal. If the probability for each class y_i, $P(y_i|\mathbf{x})$, is available, then the *Bayes* optimal prediction for $\mathbf{x}$ is the class y^* that minimises the conditional risk [Duda et al., 2012]:

$$y^* = \arg\min_{y \in \mathcal{Y}} R(y|\mathbf{x}) = \arg\min_{y \in \mathcal{Y}} \sum_{i=1}^{K} P(y_i|\mathbf{x}) C(y, y_i) \tag{2.8}$$

This provides a cost-sensitive method that solely depends on the quality of the class probability estimates. Two steps are required in this strategy: the estimation of the class probabilities and the final labelling decision. One advantage of this strategy, which was pointed by Margineantu [2002], concerns the independence between the two described steps. This means that no information of the cost matrix is used for obtaining the probability estimates and therefore, when the cost matrix changes, it is only necessary to evaluate the final decision.

Table 2.1: An example of an arbitrary cost matrix for a binary classification problem.

		True	
		$y = 0$	$y = 1$
Predicted	$\hat{y} = 0$	c_{00}	c_{01}
	$\hat{y} = 1$	c_{10}	c_{11}

We must highlight that, sometimes, in the cost-sensitive literature the cost matrix has a different definition. In fact, several authors consider that the cost matrix entries may be set to any real value, including the entries for accurate predictions. The rational behind this way of defining the cost matrix is related with, for instance, the need to perform some "expensive" tests when a given class is predicted. Therefore, it is possible to assign costs to accurate predictions. Let us term these matrices as **arbitrary cost matrices**. In this

setting, Table 2.1 may represent the cost matrix associated with a binary classification problem.

For now, let us assume the setting of arbitrary cost matrices. We will describe some of the expected properties of cost matrices that are always satisfied with our proposed definition (cf. Definition 2.3.4). Then, we show the equivalence between addressing a cost-sensitive task with our proposed definition of cost matrix and that of arbitrary cost matrices.

Some important properties of cost matrices were described by Elkan [2001] and Margineantu [2001].

Definition 2.3.6 (Row Dominance) *Let C be a cost matrix. A row m dominates a row n if, for all j, $c_{mj} \geq c_{nj}$.*

Considering the definition proposed for cost matrix (cf. Definition 2.3.4), there will never occur a row dominance situation. This is trivially verified because accurate predictions ($c_{ii} = 0$) are always set to zero, while a positive value is assigned to misclassified cases ($c_{ih} > 0$). Therefore, no row of the cost matrix can dominate another. Still, this may occur when any real values are considered as possible entries for the cost matrix. Tables 2.2 and 2.3, provided by Margineantu [2000], show two situations where this happens.

Table 2.2: Cost matrix for which one class should never be predicted.

		True		
		$y = 1$	$y = 2$	$y = 3$
	$\hat{y} = 1$	2.0	3.2	4.5
Predicted	$\hat{y} = 2$	1.0	0.1	1.0
	$\hat{y} = 3$	1.5	1.2	0.3

Table 2.3: Cost matrix for which no learning is necessary.

		True		
		$y = 1$	$y = 2$	$y = 3$
	$\hat{y} = 1$	2.0	3.2	4.5
Predicted	$\hat{y} = 2$	1.0	0.1	1.0
	$\hat{y} = 3$	1.5	2.2	1.3

In Table 2.2, we observe that the first row dominates the remaining rows, i.e., $c_{1j} > c_{2j}$ and $c_{1j} > c_{3j}$ for all $j \in \{2, 3\}$. This means that to predict class 1 has always a higher cost than to predict any other class. Therefore, we can conclude that class 1 should never be predicted as it is never optimal.

Table 2.3 represents an even more extreme situation: no learning is necessary because the optimal prediction is trivial. In this case, we observe that class 1 dominates the remaining classes and class 3 also dominates class 2. This means that class 2 has always the lowest possible costs when compared against the other classes. Therefore, to predict class 2 will always provide the lowest cost independently of this being an accurate prediction or not, i.e., for any values of $P(j = 1|\mathbf{x})$, $P(j = 2|\mathbf{x})$ and $P(j = 3|\mathbf{x})$ predicting class 2 always provides the minimum cost.

Elkan [2001] proposed the definition of reasonable conditions which guarantee that these less interesting situations do not happen. These conditions essentially state that the cost of accurate predictions should always be lower than the cost of misclassified cases.

Definition 2.3.7 (Reasonableness Conditions) *Let C be a cost matrix. The cost of accurately labelling an example is always lower than the cost of labelling it incorrectly, i.e., for all lines i of C,*

$$c_{ii} < c_{ij}, \ \forall j \tag{2.9}$$

If a cost matrix satisfies the reasonableness conditions, then no row of the cost matrix dominates any other.

Another important property of cost matrices proposed by Margineantu [2001] is related with the notion of equivalence between matrices.

Definition 2.3.8 (Equivalent Cost Matrices) *Let C_1 and C_2 be two cost matrices, and let h_1 and h_2 be two classifiers. Let $E_{C_1}(h)$ ($E_{C_2}(h)$) represent the expected classification cost of classifier h for the cost matrix C_1 (C_2). The two cost matrices are equivalent ($C_1 \equiv C_2$) if and only if for any two classifiers h_1 and h_2:*

$E_{C_1}(h_1) > E_{C_1}(h_2) \Leftrightarrow E_{C_2}(h_1) > E_{C_2}(h_2)$,

and

$E_{C_1}(h_1) = E_{C_1}(h_2) \Leftrightarrow E_{C_2}(h_1) = E_{C_2}(h_2)$.

This means that two cost matrices are equivalent when using either of them leads to the same decisions for any two classifiers.

Theorem 2.3.9 (Margineantu [2001]) *Let C_1 be an arbitrary cost matrix. Let Δ be a matrix of the form:*

$$\Delta = \begin{bmatrix} \delta_1 & \delta_2 & \delta_3 & \dots & \delta_k \\ \delta_1 & \delta_2 & \delta_3 & \dots & \delta_k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \delta_1 & \delta_2 & \delta_3 & \dots & \delta_k \end{bmatrix}$$

If $C_2 = C_1 + \Delta$*, then* $C_1 \equiv C_2$*.*

Corollary 2.3.9.1 (Margineantu [2001]) *An arbitrary cost matrix can always be transformed into an equivalent matrix with zero on the diagonal or into an equivalent matrix with non-negative values.*

Corollary 2.3.9.1 shows that considering a cost matrix with zero cost for accurate predictions is equivalent to other arbitrary matrices with non-zero values on the diagonal. Therefore, all the solutions proposed for arbitrary cost matrices are also valid for cost matrices defined as we propose. Without loss of generality, we will assume the proposed cost matrix definition with zero cost for accurate predictions.

Cost-sensitive learning has been explored mostly for the binary class case as this has shown to be an easier to handle setting in comparison with the multiclass scenario. For this reason, the theoretical developments are also more focused on the binary case.

Elkan [2001] studied, for the two-class case, the optimal decision threshold for making predictions. Let us consider a classification task with two classes ($y \in \{0, 1\}$). Typically, in two class problems, the minority or rare class is considered the positive class. Let us suppose that the positive class has label 1. Let p represent the conditional probability that an example $\mathbf{x}$ is classified as belonging to class 1, i.e., $p = P(y = 1|\mathbf{x})$. Then, the conditional probability for the example belonging to class 0 is $1 - p$. The optimal prediction for case $\mathbf{x}$ is the class 1 if and only if the expected classification cost of predicting class 1 is lower than that of predicting class 0, i.e.,

$$(1 - p) \cdot c_{10} + p \cdot c_{11} < (1 - p) \cdot c_{00} + p \cdot c_{01} \tag{2.10}$$

When the above inequality becomes an equality, the expected cost of both classes is the same, and therefore, predicting either class for example $\mathbf{x}$ is optimal. The inequality leads to the following theoretical threshold for making optimal decisions:

$$p^* = \frac{c_{10} - c_{00}}{c_{10} - c_{11} + c_{01} - c_{00}} \tag{2.11}$$

When considering that accurate predictions have zero cost ($c_{ii} = 0$), the theoretical threshold can be simplified to

$$p^* = \frac{c_{10}}{c_{10} + c_{01}} \tag{2.12}$$

Therefore, predicting the positive class (class 1) is optimal whenever $p \geq p^*$.

This theoretical threshold can be used for cost-sensitive learning when the selected classifier is able to output a posterior probability estimation for each test instance.

Still, some classifiers are not capable of producing such probabilities. For this case, Elkan [2001] showed how to rebalance the training set by sampling, and provided the optimal way of changing the proportion of positive and negative class examples (cf. Theorem 2.3.10).

Theorem 2.3.10 (Elkan [2001]) *To make a target probability threshold p^* correspond to a given probability threshold p_0, the number of negative examples in the training set should be multiplied by*

$$\frac{p^*}{1-p^*} \cdot \frac{1-p_0}{p_0} \tag{2.13}$$

Considering cost matrices with zeros in the diagonal and the most frequently used threshold of 0.5, Theorem 2.3.10 states that the number of negative examples should be multiplied by $\frac{p^*}{1-p^*} = \frac{c_{10}}{c_{01}}$. This is a special case that was also used by Breiman et al. [1984].
Therefore, the theoretical proportion between the number of positive (n_+) and negative (n_-) class examples in the training set is as follows:

$$\frac{n_+ \cdot c_{01}}{n_- \cdot c_{10}} \tag{2.14}$$

In a binary class problem, also the class prior probabilities (p_+ and p_-) and costs are interchangeable, and, for instance, doubling p_+ will have the same impact as doubling c_{01} [Drummond and Holte, 2000c, Provost et al., 1998, Provost and Fawcett, 1997].

The theoretical advances for the multiclass case have been more scarce. In fact, frequently the adopted strategy is to simply reduce the multiclass task to several binary class tasks using different strategies to achieve this conversion. A relevant result was presented by Zhou and Liu [2010] regarding the existence of an optimal way of rescaling multiclass cost-sensitive problems. Although the theoretical proportion has shown to be efficient for two class problems its efficiency when dealing with multiclass problems has been poor. Zhou and Liu [2010] derived a property of cost matrices named consistency (cf. Definition 2.3.11) that allows to determine whether it is possible or not to optimally rescale the problem classes. The optimal rescale of classes is only possible when the cost matrix is consistent. When the cost matrix does not have this property it is possible to select another method which will necessary be suboptimal.

Definition 2.3.11 (Consistent cost matrix) *Let C be a cost matrix representing a classification problem with k classes. C is a consistent matrix when matrix C', derived from C*

as show below, has lower rank than k.

$$C' = \begin{bmatrix} c_{21} & -c_{12} & 0 & \dots & 0 \\ c_{31} & 0 & -c_{13} & \dots & 0 \\ \dots & \dots & \dots & \dots & 0 \\ c_{k1} & 0 & 0 & \dots & -c_{1k} \\ 0 & c_{32} & -c_{23} & \dots & 0 \\ \dots & \dots & \dots & \dots & 0 \\ 0 & c_{k2} & 0 & \dots & -c_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & -c_{(k-1)k} \end{bmatrix}$$

Cost-sensitive learning is the most popular framework for non-standard learning tasks. Still, the impact of the predictions in a given domain should not be restricted to costs. In fact, although incorrect predictions may lead to costs, the accurate predictions may also imply profits or benefits. Therefore, to only consider costs may be too restrictive for several real world problems. The broader notion of utility should be adopted and its integration on the different data mining tasks must be considered [Weiss et al., 2005, Zadrozny et al., 2006]. Moreover, as pointed by Elkan [2001], the consideration of both benefits and costs (negative benefits) has also the advantage of providing a more intuitive baseline. The baseline that must be considered is the state of the decision maker before a decision is taken for a given example. After the decision is made, if it implies a positive change in the state we have a benefit, otherwise, we have a cost (negative benefit). According to Elkan [2001], it is easier to avoid mistakes when this setting is considered.

Definition 2.3.12 (Utility Matrix) *Consider a classification task with K different classes. A utility matrix $U := [u_{ij}]$ specifies the benefits and costs of predictions. More precisely, it sets the benefits obtained when an example is accurately predicted, and the costs incurred when an example is misclassified. This matrix can be more formally defined as follows:*

$$\begin{aligned} U : K \times K &\rightarrow \mathbb{R} \\ (i,j) &\rightarrow u_{ij} = \begin{cases} \geq 0 & i = j \\ < 0 & i \neq j \end{cases} \end{aligned} \tag{2.15}$$

where u_{ij} provides the utility of predicting class i for an example with true class j.

The notion of utility matrix is an extension of the cost matrix concept where the entries can have both positive and negative values. Moreover, the diagonal elements, corresponding to accurate predictions, must have non-negative values while the remaining elements should have negative values. When building a utility matrix, any baseline can be considered for measuring costs and benefits, as long as it is fixed [Elkan, 2001].

In this setting, the goal of the predictive task is the maximisation of the expected utility.

Definition 2.3.13 (Expected Classification Utility) *Given a utility matrix U, the expected utility of a classifier h in an instance* $\langle \boldsymbol{x}, y \rangle \in \mathcal{X} \times \mathcal{Y}$ *drawn from a distribution* $\mathcal{D}$ *is defined as*

$$E_{CU}(h, \langle \boldsymbol{x}, y \rangle, \mathcal{D}) := \mathbb{E}_{\mathcal{D}}[U(h(\boldsymbol{x}), y)] \tag{2.16}$$

As we previously mentioned, in a classification problem, to consider a cost matrix with arbitrary entry values is equivalent to consider a cost matrix with zeros in the diagonal (cf. Corollary 2.3.9.1). This means that it is always possible to transform the cost matrix defined for a given problem into an equivalent cost matrix with zero costs for accurate predictions. Therefore, the problem of minimising the costs with both cost matrices is equivalent. In a utility setting, the goal is different as it involves the maximisation of the utility. Still, if the utility matrix is multiplied by -1, then, the problem of maximising the utility becomes equivalent to the problem of minimising the symmetric of the utility matrix. Moreover, this new symmetric matrix, containing negative values in the diagonal and positive values in the off-diagonal entries, is equivalent to a matrix with zeros in the diagonal. This means that, the problem of minimising the costs is equivalent to the problem of maximising the utility. Let us see an example of this equivalence using the calling card fraud detection problem presented by Margineantu [2001].

Example 2.3.14 (Calling Card Fraud Detection Problem) *Let us consider the cost matrix in Table 2.4. This matrix represents the costs incurred into when a call is labelled as Fraud or No-Fraud by the decision maker. If a given call is predicted as fraudulent, the consequence is the immediate close of the account associated with the call. If the call is predicted as non-fraudulent, then no action is taken.*

The matrix displays a high cost for the agent when a fraudulent call is misclassified as legal. When a legal call is mislabelled as fraudulent it is necessary to take into account both the costs related with closing and reopening the account and the client dissatisfaction. Calls accurately predicted as fraud have the benefit of protecting the clients although they also assume the cost of closing the account.

This matrix can be transformed into an equivalent one with zeros in the diagonal as displayed in Table 2.5. The problem of minimising the costs using either of the cost matrices is equivalent.

Let us now consider the utility matrix defined in Table 2.6. This matrix is obtained by multiplying by -1 the cost matrix in Table 2.4. The problem of minimising the costs using the matrix in Table 2.4 is equivalent to the problem of maximising the utility using the matrix in Table 2.6.

Until this moment we have only considered stationary cost and utility matrices, i.e., matrices that are known before the model is built and that do not change. However, this assumption is

Table 2.4: Cost matrix proposed by Margineantu [2001] for the calling card fraud detection problem.

		True	
		Fraud	No-Fraud
Predicted	Fraud	-1.5	10
	No-Fraud	100	0

Table 2.5: Transformed cost matrix for the calling card fraud detection problem.

		True	
		Fraud	No-Fraud
Predicted	Fraud	0	10
	No-Fraud	101.5	0

Table 2.6: Transformed utility matrix for the calling card fraud detection problem.

		True	
		Fraud	No-Fraud
Predicted	Fraud	1.5	-10
	No-Fraud	-100	0

often considered unrealistic in real world problems [Provost and Fawcett, 2001, Margineantu, 2001]. This means that a more complex setting should be considered which allows that errors and accurate predictions of the same type may incur into different cost/utility values. This can happen in different ways. For instance, the matrix entries may be time dependent, may be example dependent or may express a more complex dependence on other factors such as the trigger of an event. A cost/utility matrix that allows that the entries defined change based on some factors is named a dynamic cost/utility matrix. Dynamic cost and utility matrices are a generalisation of stationary matrices. Table 2.7 shows an example of a stationary and a dynamic cost matrix for a two-class problem. In the former the matrix entries remain unchanged while in latter the entries change according to the definition of f_{01} and f_{10} functions. These functions can represent a diversity of problem settings. For instance, they can represent the monetary cost associated with each example (e.g. Bahnsen et al. [2015]), can be associated with distance measurements when the examples have a spatial component (e.g. Burl et al. [1998]) or can even be a function of time when the data has this information (e.g. Fawcett and Provost [1999]).

Dynamic cost and utility matrices are meaningful in many contexts. For instance, for the

Table 2.7: Example of a stationary cost matrix (left) and a dynamic cost matrix (right) for a two-class problem.

		True	
		$y = 0$	$y = 1$
Predicted	$\hat{y} = 0$	0	3
	$\hat{y} = 1$	10	0

		True	
		$y = 0$	$y = 1$
Predicted	$\hat{y} = 0$	0	$3 \cdot f_{01}$
	$\hat{y} = 1$	$10 \cdot f_{10}$	0

cost-sensitive problem of credit card fraud (e.g. credit card cloning) the costs increase with the time elapsed before the fraud is detected. In this case, the longer the situation remains undetected, the higher are the losses. Another example can be found in activity monitoring tasks. In this context, which involves time-dependent applications, Fawcett and Provost [1999] highlighted that the costs are typically overestimated because the costs should only be considered until the first true positive alarm. This means that, after the event is triggered by a true positive alarm, the costs are meaningless because the situation was already detected.

As mentioned before, the problem of utility-based learning can be thought as a generalisation of the cost-sensitive learning problem. As such, we will, from this point on, simply use the term utility-based learning as an aggregating concept that incorporates both approaches because one problem can be reduced to the other.

The problem of utility-based learning has also been considered for regression tasks [Torgo and Ribeiro, 2007]. In this setting, the information regarding both costs and benefits is provided for all pairs of values $(\hat{y}, y)$ through a utility surface, given that the domain of the target variable is continuous.

Definition 2.3.15 (Utility Surface) *Consider a regression problem with target variable domain $\mathcal{Y}$. A utility surface $U := u(\hat{y}, y)$ specifies the utility (benefit or cost) for predicting $\hat{y}$ for a true target variable value of y. More precisely, it maps each point $(\hat{y}, y)$ into a utility score, where negative utility values correspond to the costs incurred and non negative values correspond to benefits achieved.*

A utility surface can be interpreted as a continuous version of a utility matrix. However, to fully specify this surface is usually too challenging for the end-user. Two main solutions have been put forward: (i) interpolate the surface based on a few points supplied by the user; (ii) automatic derivation of the surface that is possible for some particular sub-classes of utility-based regression tasks.

The first approach is simple and more generally applicable. The user supplies a few values of the utility surface for carefully selected pairs of $\hat{y}$ and y values. A standard function interpolation method is then used to obtain the full surface.

The second approach was proposed by Ribeiro [2011] and can be applied under some assumptions concerning the user preferences. More specifically, this method can be applied when the goal of the end-user is to obtain a model that is good at forecasting rare extreme values of the target continuous variable. This is a particular case of imbalanced regression tasks. Although specific, this setting is actually rather frequent in several real world applications like forecasting extreme values in several ecological monitoring tasks, or forecasting stock market returns.

Given that utility-based regression is one of the main topics of this thesis further details on these and other methods will be presented in the next chapters.

2.3.2 Performance Assessment in Utility-based Learning

Performance assessment is an important challenge in utility-based learning, as mentioned in Section 2.2. In these contexts, to simply analyse the scores of a loss function is insufficient because it is necessary to take into account the domain information provided through a utility matrix or surface. Standard loss functions may be misleading because they do not take into account the end-user preferences.

Performance assessment metrics may be categorised into scalar-based and graphical-based. In the former, the results are presented as a single number, while in the latter, the results are displayed graphically or are derived from those representations. Scalar metrics are practical because they are a succinct representation of the performance. However, they also present important drawbacks because they may discard relevant information. In particular, these metrics are suitable for cases where the user knows in advance the deployment scenario of the learned model. However, if that is not the case, then graphical-based metrics may be more adequate [Japkowicz, 2013]. Graphical-based metrics allow to observe the performance of a model across all possible operating conditions, which represents an advantage when the deployment scenario is unknown. Table 2.8 summarises the main references and metrics for performance assessment in utility-based learning tasks.

Scalar Metrics Considering the cost-sensitive framework, where the goal is the minimisation of costs, the expected cost provides the baseline for performance assessment. A typically used measure is the total cost, TC, which corresponds to summing all the predictions costs (e.g. Elkan [2001]). Several measures can be derived such as the Mean Cost (MC). Given a set of N cases this metric is defined as follows:

$$MC = \frac{1}{N} \sum_{i=1}^{N} C(\hat{y}_i, y_i) \tag{2.17}$$

Table 2.8: Performance assessment metrics and main bibliographic references for utility-based learning tasks.

Metric Type	Task type	Metric	Main References
Scalar	Classification	TC, MC, NMC	Elkan [2001], Turney [1995]
	Regression	$LIN - LIN$, $QUAD - QUAD$, $LIN - EXP$, $QUAD - EXP$, MU	Zellner [1986], Cain and Janssen [1995], Christoffersen and Diebold [1996, 1997], Crone et al. [2005], Granger [1999], Lee [2008], Ribeiro [2011]
Graphical	Classification	*Cost curves*	Drummond and Holte [2000b]
	Regression	*REC curves*, *RROC space*, *AOC*	Bi and Bennett [2003], Hernández-Orallo [2013]

Turney [1995] proposed the Normalised Mean Cost (NMC) metric for classification due to several concerns related with the use of MC metric. These concerns were related with the difficulty in comparing in a fair way different data sets, and matrices with different magnitudes of costs. The new NMC measure was defined as a ratio between the MC and the standard cost as follows:

$$NMC = \frac{MC}{T + min_i(1 - f_i) \cdot max_{i,j}(c_{ij})} \tag{2.18}$$

where T represents the total cost of doing all of the possible tests, f_i is the fraction of cases in the data set that belong to class i, and c_{ij} is the cost of predicting class i for a true class j. Although the standard cost considered (denominator in Equation 2.18) does not represent the worst possible scenario, Turney [1995] claimed that, based on the experiments conducted, the standard cost is better for normalisation providing a more realistic upper bound on the average cost.

In regression tasks, one of the first attempts to deal with non-uniform preferences emerged in the financial area with the proposal of asymmetric loss functions [Zellner, 1986, Cain and Janssen, 1995, Christoffersen and Diebold, 1996, 1997, Crone et al., 2005, Granger, 1999, Lee, 2008]. Such functions, allow the consideration of differentiated prediction costs for two specific types of errors: under and over predictions. The $LIN - LIN$ (asymmetric linear on both sides) or the $QUAD - EXP$ (approximately quadratic on one side and exponential on the other) are examples of asymmetric loss functions. However, all these solutions suffer from the same problem: they only allow to distinguish between under- and over-predictions and do not take into account the location where the error occurred. This may not adapt to the user preferences that may be biased towards specific ranges of the target variable rather than to assigning a different penalisation to over- and under-predictions.

When having the full utility surface specified, the goal is utility maximisation and the natural extension to a utility setting of the MC measure defined for classification, leads to the Mean

Utility (MU) (cf. Equation 2.19), proposed by Ribeiro [2011].

$$MU = \frac{1}{N}\sum_{i=1}^{N} U(\hat{y}_i, y_i) \tag{2.19}$$

The use of MU allows characterising the performance of different models taking into account the end-user preference biases. Still, the use of MU may not be suitable for comparing the performance of models across different data sets because the different utility matrices or surfaces may be defined in different scales. The use of a normalised metric allows to overcome this problem making the comparison between different data sets easier.

Graphical-based Metrics An interesting graphical performance assessment tool, suitable for binary class problems, are cost curves, that were introduced by Drummond and Holte [2000b]. Cost curves represent the expected cost in the y-axis, while the x-axis displays the probability cost function. Both axis are normalised to the interval $[0, 1]$. Equation 2.20 shows the probability cost function definition,

$$PCF(c_1) = \frac{p(c_1)C(c_2, c_1)}{p(c_1)C(c_2, c_1) + p(c_1)C(c_1, c_2)} \tag{2.20}$$

where $p(c_1)$ represents the probability of a given class c_1 and $C(c_1, c_2)$ represents the cost of misclassifying an example of a class c_2 as being of class c_1.

Figure 2.1 displays an example of a cost curve for the Pima Indians[1] data set.

The popularity of Receiver Operating Characteristics curve (ROC) curves in classification motivated several attempts to extend this concept to a regression context. The ROC Space for Regression (RROC Curve) [Hernández-Orallo, 2013] is the result of one of those attempts. Behind RROC curves there is the assumption that only over- and under-predictions entail different costs. A point in the RROC Curve corresponds to plotting the total over- and under-estimation on the x-axis and y-axis, respectively, as shown in Figure 2.2). The RROC Curve is obtained by applying a shift to the predictions, i.e., by adding or subtracting a constant to the predictions which allows to adjust the model to an asymmetric operating condition.

The Area Over the RROC Curve (AOC) is a single value metric, proposed by [Hernández-Orallo, 2013], that is derived from the RROC curves and aggregates a models performance on different operating conditions. AOC metric was shown to be equivalent to the error variance.

Another interesting graphical tool is the Regression Error Characteristic Curves (REC

[1] Pima Indians Diabetes Database data set is available through UCI Repository [Dheeru and Karra Taniskidou, 2017] at `https://archive.ics.uci.edu/ml/datasets/pima+indians+diabetes`.

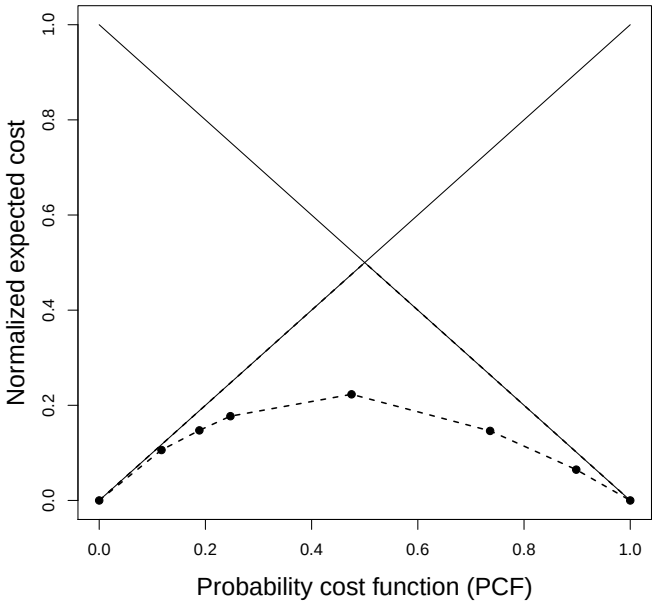


Figure 2.1: Example of a Cost Curve for the pima data set.

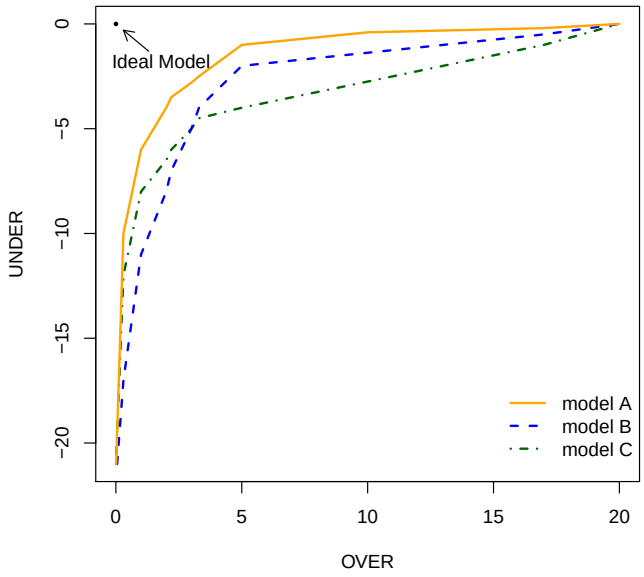


Figure 2.2: RROC Curve of three models: A, B and C.

Curves) proposed by Bi and Bennett [2003]. REC Curves present the cumulative distribution function of the error of a model. The strategy followed to show this is to plot the error tolerance and the accuracy of a regression function which is defined as the percentage of points predicted within a given tolerance ϵ. This means that REC Curves are able to display the performance of a model for the range of possible errors. Figure 2.3 shows an example of a REC Curve.

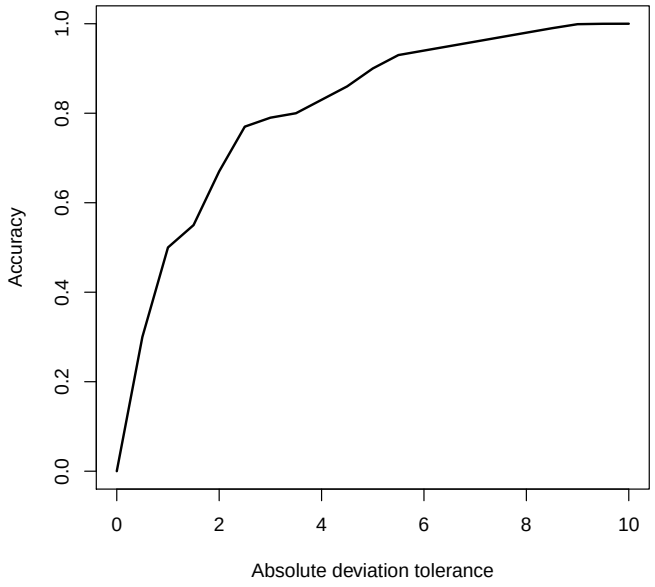


Figure 2.3: An example of a REC Curve.

2.3.3 Learning Methods for Utility Optimization

The research community has proposed a diversity of methods for learning models that maximize utility. As we have mentioned, they can be grouped in the two following main types [Ling and Sheng, 2011a]: direct methods, and meta-learning methods. The former change the learning algorithm internally while the latter are "wrapper" methods that transform any standard utility-insensitive learner into a utility-sensitive one. Meta-learning approaches allow the use of any standard learner, because the methods either pre-process the training data or post-process their predictions. Ling and Sheng [2011a] proposed to further categorise these latter methods into: thresholding and sampling. Figure 2.4 shows a taxonomy for utility-based learning strategies proposed by Ling and Sheng [2011a].

Table 2.9 provides a brief overview of these methods displaying a summary of the main advantages and disadvantages of each type of utility-based learning strategy. In the two following subsections we discuss with more detail the two main types of methods.

Table 2.9: Main advantages and disadvantages of each type of strategy for utility-based learning.

Strategy	Advantages	Disadvantages
Direct Methods	• costs/benefits are incorporated directly into the models • models obtained are more comprehensible to the user • some methods that directly incorporate misclassification costs/benefits when building the model can also easily integrate other types of costs/benefits	• if the utility matrix changes, the model needs to be relearned • user is restricted in his choice of the learning algorithms that have been modified to incorporate costs/benefits • the adaptation of a learner is difficult as it requires a deep knowledge of the learning algorithms implementations
Meta-learning Methods: Thresholding	• the utility matrix information is not used during the learning phase, therefore the learned models can be applied to different deployment scenarios without the need of re-learning the models or even keeping the training data available	• the models do not reflect the user preferences • the selected learner must output probabilities • the strategies depend on accurate probability estimations, which may not be easy to obtain • generally, only misclassification costs are considered, being hard to consider other types of costs • models interpretability may be jeopardised as they were obtained by optimising a loss function that is not in accordance with the utility scenario used
Meta-learning Methods: Sampling	• models are biased towards the user preferences expressed through the defined utility matrix • any standard learning tool can be used	• generally, only misclassification costs are considered, being hard to consider other types of costs • difficulty of relating the modifications in the data distribution with the costs

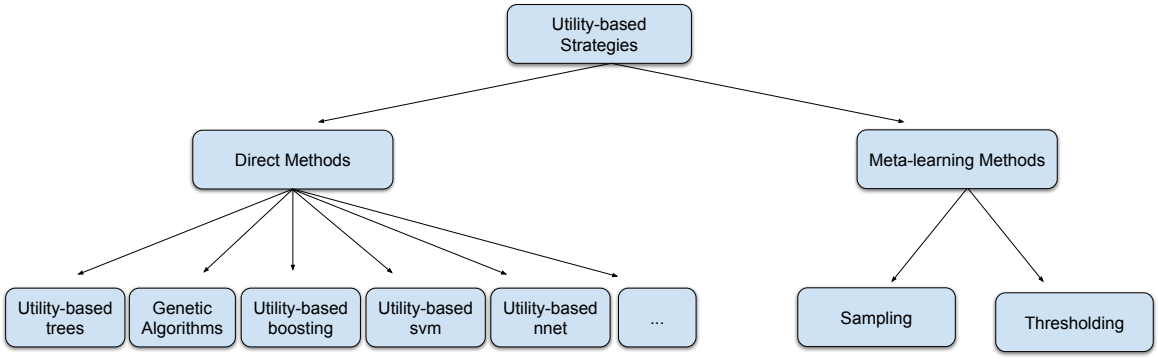


Figure 2.4: Taxonomy proposed by Ling and Sheng [2011a] for utility-based strategies.

2.3.3.1 Direct Methods

The key idea of direct methods is to change a given learning algorithm (or to develop a new learning algorithm) in order to make it capable of internally dealing with the utility information. This requires a thorough knowledge of the learning algorithm implementation to achieve the integration of costs and benefits in the learning process.

Several works exist on direct methods for utility-based learning including as base learners genetic algorithms, decision trees, Support Vector Machines (SVMs), Neural Networks (NNETs), bagging or boosting (bootstrap aggregating). We will briefly present the main methods for decision trees and ensembles, with other important direct methods summarised in Table 2.10.

Decision Trees. A large number of proposals incorporate cost information in decision trees. Figure 2.5 shows the taxonomy proposed by Lomax and Vadera [2013] for cost-sensitive decision tree algorithms. This taxonomy distinguishes between the methods that train a single or multiple decision trees. The first can be further clustered into those that use costs during the tree construction and the ones that apply the costs after the tree is built. We consider the latter as meta-learning methods and will describe them later in this chapter.

A possible way for using costs during the decision tree construction is to change the splitting criterion. A new measure that is able to take into account the cost information can be used for selecting the split tests on each tree node. Several algorithms were proposed that follow this intuition (e.g. Ailing et al. [2005], Zhang et al. [2007]). Many of these algorithms can also deal with costs associated with the predictors, such as, costs for testing the value of a variable in a split node.

Breiman et al. [1984] proposed an alternative approach that changes the class probabilities that are used in the Information Gain splitting criterion. Pazzani et al. [1994] applied the

idea of altering the class probabilities to the GINI index for building cost-sensitive decision trees.

During tree construction, an alternative possibility is to use directly the misclassification costs as the splitting criterion (e.g. Pazzani et al. [1994], Ling et al. [2004, 2006]). The key idea of this strategy is to select a split on the attribute that produces the higher reduction of the expected cost. If no split allows this reduction, then, a leaf node is created.

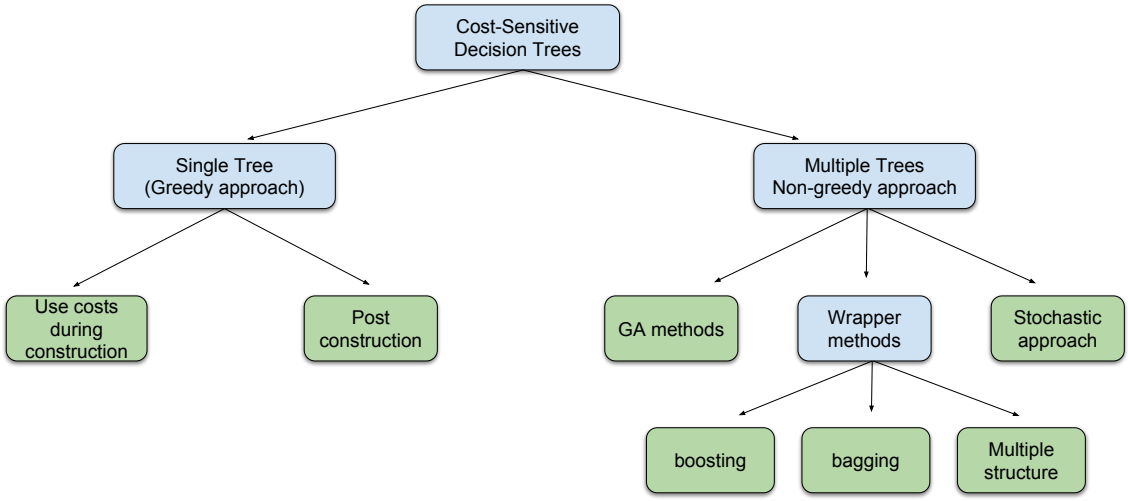


Figure 2.5: Taxonomy for cost-sensitive decision trees algorithms [Lomax and Vadera, 2013].

Ensembles. Ensemble methods have been gaining attention over the last years being known to improve the performance of learning algorithms by combining several of them [Galar et al., 2012]. The key idea of ensemble learning is to build multiple models using the available training data and then aggregate their predictions on unseen examples through a certain mechanism. Many ensemble methods were also adapted for being able to deal with cost information. We will discuss the adaptation to a cost framework of bagging and boosting.

The key idea of bagging [Breiman, 1996] is the generation of a set of models using different bootstrap samples of the training set. The final predictions are then obtained by averaging or voting. The bagging method is able to improve the performance of learning algorithms, in particular if the base learners used are not stable [Breiman, 1996].

Lin and McClean [2000] developed a bagging method with a final voting scheme based on a risk measure proposed by the authors. This new risk measure uses the prior probabilities of the examples, the defined costs and the predictions of all trained learners.

Margineantu [1999] proposed an algorithm for cost-sensitive bagging in the context of multiclass tasks. The algorithm incorporates the cost information by voting the predictions of decision trees grown using bootstrap samples of weighted examples. Several other bagging-

based proposals were put forward (e.g. Moret et al. [2006], Zhou and Liu [2006]).

Boosting [Schapire, 1990, Freund, 1995] is an iterative algorithm that trains weak learners adding them in a weighted, sequential fashion to build the final model. Each training sample is adaptively weighted at each boosting round, being higher weights assigned to examples most often misclassified. One of the first and most successful boosting methods is AdaBoost (Adaptive Boosting) [Freund and Schapire, 1995]. For incorporating cost information in boosting it is necessary to exchange its goal from minimising the errors in all classes to obtaining an unequal treatment of classes guided by the costs defined for the problem. This can be achieved by changing: the algorithm initialisation, the weight update rule or the final weighted combination of hypothesis. Several approaches exist for cost-sensitive boosting. Many of them are based on altering the AdaBoost method by simply changing the weight update rule. They essentially differ in the way they incorporate costs into this formula. Examples of such proposals are: AdaCost [Fan et al., 1999], CSB1 and CSB2 algorithms [Ting, 2000a], AdaC1, AdaC2 and AdaC3 [Sun et al., 2007], UBoost and Cost-Uboost [Ting and Zheng, 1998], Loss Minimization Boosting [Margineantu, 1999] or Assymetric Boosting [Masnadi-Shirazi and Vasconcelos, 2007].

The application of boosting to a multiclass context is harder because an example can be misclassified into different classes and therefore the determination of weights is not trivial. Several methods were proposed to address this problem (e.g. Breiman et al. [1984], Margineantu [2001], Abe et al. [2004]).

There are also proposals that integrate boosting with other base learners. This is the case for the work of Zheng [2010] where an algorithm for boosting NNETs is presented.

The work in the context of regression tasks is scarce. Ribeiro [2011] proposed a utility-based ensemble system named utility-based Rules (ubaRules). The method is designed for obtaining models that are biased towards a specific utility function. The key goal of ubaRules is to obtain predictions both accurate and interpretable for regression tasks with a non-uniform utility. ubaRules involves two main steps: i) grow different regression trees and convert them to an ensemble of rules; ii) select the more suitable rules to include in the final ensemble. The utility information is used at different levels of the algorithm.

2.3.3.2 Meta-learning Methods

Meta-learning methods transform any standard cost insensitive learner into a cost-sensitive one without modifying it. Meta-learning methods may be further categorised into thresholding and sampling methods. The first group is related with changing the decision threshold to classify examples when the learner outputs probabilities. The sampling methods modify the target variable distribution on the training data. After this step, any out-of-the-box

Table 2.10: Main bibliographic references of works optimising different base learning algorithms for utility maximisation chronologically ordered.

Base Learner	**Main References**
decision tree	Breiman et al. [1984], Pazzani et al. [1994], Bradford et al. [1998], Ting and Zheng [1998], Margineantu [2001], Ting [2002], Ling et al. [2004], Ailing et al. [2005], Ling et al. [2006], Zhang et al. [2007]
NNET	Kukar and Kononenko [1998], Wan et al. [1999], Zhou and Liu [2006]
SVM	Amari and Wu [1999], Karakoulas and Shawe-Taylor [1999], Fumera and Roli [2002], Lin et al. [2002], Kandola and Shawe-Taylor [2003], Wu and Srihari [2003], Wu and Chang [2003a], Akbani et al. [2004], Geibel et al. [2004], Wu and Chang [2005], Bach et al. [2006], Masnadi-Shirazi et al. [2012]
Genetic Algorithm (GA)	Turney [1995], Li et al. [2005], Kretowski and Grześ [2007], Omielan and Vadera [2012]
boosting	Breiman et al. [1984], Ting and Zheng [1998], Fan et al. [1999], Margineantu [1999, 2001], Ting [2000a], Abe et al. [2004], Sun et al. [2007], Masnadi-Shirazi and Vasconcelos [2007], Zheng [2010]
bagging	Margineantu [1999], Lin and McClean [2000], Moret et al. [2006], Zhou and Liu [2006]
random forest	Chen et al. [2004], Yang et al. [2009], Xie et al. [2009]

learner can be used directly.

Meta-learning methods have several advantages. The sampling methods allow the use of any standard learning algorithm enabling the user to select the algorithm that he thinks is more suitable for the problem being tackled. The thresholding methods allow the application of the learned models to different deployment contexts without the need of re-learning the models. However, frequently these methods are hard to apply because it may not be straightforward to determine the ideal change to apply. We will briefly present the main meta-learning methods which are summarised in Table 2.11

Thresholding. According to Ling and Sheng [2011a], the methods categorised under thresholding are based on Equation 2.12 (page 16). These methods are only applicable when the learner is able to output probabilities because they act by using the definition of a theoretical threshold on the probabilities for making decisions.

Thresholding can be used for relabelling either the training or the test instances. MetaCost [Domingos, 1999] starts by obtaining more reliable probability estimates of the training set by applying bagging on decision trees. Then, the training set classes are relabelled for minimising the conditional risk. Finally, a new cost-insensitive learner is trained on the new modified training set.

Ting [2000b] studied the use of cost-sensitive boosting algorithms as base learners in the MetaCost method. Namely, he tried the use of AdaBoost and Cost-UBoost in conjunction with MetaCost. The results showed little improvements in the minimisation of costs. Still, an advantage was observed when using a cost-sensitive algorithm as the base learner in MetaCost in comparison to using a cost-insensitive learner [Ting, 2000b, Vadera, 2010].

Generally, thresholding methods depend on reliable probability estimations. However, these accurate estimations are frequently hard to obtain. This motivated the multiple proposals made by Zadrozny and Elkan [2001] that try to improve the calibration of probability estimations. Sheng and Ling [2006] proposed an empirical thresholding method that does not need accurate probabilities estimations, requiring only an accurate ranking. This method does not use the theoretical threshold presented in Equation 2.12. Instead, this method uses cross validation to search for the best threshold on the training set. The learned threshold is then used to obtain the predictions on the test set.

Recently, thesholding has also been explored in the context of regression tasks. The reframing proposal [Hernández-Orallo, 2012, 2014, Hernández-Orallo et al., 2016] aims at adjusting the predictions to different deployment contexts. Reframing is a post-processing method, suitable for both classification and regression problems, that modifies the obtained predictions making them adequate to a new context. The reframing method consists of two main steps:

- convert a standard model into a new model that can be seen as a conditional density

estimator, by using enrichment methods;

- the reframing of the new model to new contexts, that is achieved through an instance-dependent optimisation procedure.

Sampling The methods categorised under this section are related with the use of Equation 2.14 (page 17). The main idea is to change the training data distribution and then any standard learner can be applied on the modified training set.

Zadrozny et al. [2003] showed that applying sampling with replacement may cause an overfitting problem due to the introduction of exact replicas in the training data. As such, Zadrozny et al. [2003] proposed the *costing* method to deal with this problem that is based on the notion of "rejection sampling" for avoiding duplicate cases in the train set. The key idea is to obtain a new training set where each example is selected for being included with a probability proportional to its associated weight. These weights can be either provided for each training example, or be calculated based on the cost matrix.

The weighting method [Ting, 1998] can also be considered a sampling approach. This method associates a weight to each case according to the misclassification costs expressed in Equation 2.14. After the weights assignment, any learner that is able to internally handle examples weights can be used.

Sampling methods are more frequently used for handling imbalanced domains, which as we have seen are a special case of utility-based learning tasks. In Section 2.4.3.2, we describe in more detail these methods.

Other Meta-learning Methods Some methods use other strategies for handling cost-sensitive learning, that nevertheless can be seen as belonging to the meta-learning category. For instance, when learning a decision tree, the labels of the leaves may be changed after the tree is grown. This can be performed if, for instance, the information regarding the costs is unknown when the tree is grown, or even if the cost matrix is expected to change. Pazzani et al. [1994] proposed the I-gain Cost-Laplace Probability method. An example is assigned to the class that minimises the expected cost of misclassification that is evaluated by taking into account both the cost matrix entries and the Laplace corrected probabilities. Ferri et al. [2002] proposed the AUCSplit method that uses the area under the ROC curve to post-process the labels of the leaves of a decision tree.

For regression tasks, Bansal et al. [2008] proposed a method for cost sensitive regression that changes the predictions of a cost insensitive model. The new predictions are determined by a polynomial function that is applied to the original predicted values. The coefficients of this polynomial function are estimated by an iterative hill-climbing algorithm. Still, this method was developed to tackle regression problems that consider asymmetric costs in terms of under

and over predictions. The more complex setting of considering a general utility function was not addressed.

Table 2.11: Main bibliographic references of meta-learning methods for utility maximisation chronologically ordered.

Strategy Type	Main References
Thresholding	Domingos [1999], Ting [2000b], Zadrozny and Elkan [2001], Sheng and Ling [2006], Vadera [2010], Hernández-Orallo et al. [2012]
Sampling	Ting [1998], Zadrozny et al. [2003]
Other	Pazzani et al. [1994], Ferri et al. [2002], Bansal et al. [2008], Zhao et al. [2011], Hernández-Orallo [2012], Hernández-Orallo et al. [2016]

2.4 Imbalanced Domains

In multiple real world applications, the preferences of the end-user are geared towards cases that are underrepresented in the available data. These tasks are known as imbalanced domains.

In the context of learning from imbalanced domains, rare cases are the most interesting for the user, and therefore, these are also the cases with higher utility. As we have mentioned, imbalanced domains are a sub-class of utility-based problems, both assuming non-uniform preferences, but the former adding a preference bias towards underrepresented cases. It is this lack of data concerning the more important cases that raises extra difficulties to learning from imbalanced domains when compared to the general utility-based learning tasks. On top of this, it is also frequent that the way these user preferences are expressed is also a bit more informal, meaning that we usually do not have access to a utility matrix or surface, but simply the information that rare cases are more important.

The problem of imbalanced domains was first studied for classification tasks, and in particular for two class problems. Therefore, the majority of solutions for this problem are still concentrated in binary classification tasks. More recently, it was shown that the problem of imbalanced domains also arises in several other tasks, namely, regression, time series, data streams or multi-label prediction tasks [Branco et al., 2016b, Krawczyk, 2016].

Imbalanced domains for regression tasks is one of the topics addressed in this thesis. In this context, the next sections will provide a broad overview of the state of the art on imbalanced domains. We start by defining the problem of learning from imbalanced domains and then describe the main solutions available in the literature.

2.4.1 Problem Definition

The problem of imbalanced domains can be defined as follows:

Definition 2.4.1 (Imbalanced Domains) *Consider a predictive task where the goal is to learn a model $h(\mathbf{x})$ using a training set $\mathcal{D}$. We face an imbalanced domain problem when the following assertions occur simultaneously:*

1. *the importance assigned by the end-user to the predictive performance of the model $h(\mathbf{x})$ is non-uniform across the domain of the target variable ($\mathcal{Y}$); and*
2. *the more relevant cases for the end-user are poorly represented in the training set $\mathcal{D}$.*

Assertion (1) means that imbalanced domains are also utility-based learning tasks. However, due to the requirement of second assertion they are a specific type of utility-based tasks. The rarity of the most important cases has consequences in terms of the learning procedures. More specifically, it creates serious challenges to the learning algorithms due to the lack of statistical support of the important cases in the available training sample.

2.4.2 Performance Assessment in Imbalanced Domains

Standard evaluation metrics are unsuitable in the context of imbalanced domains as their use may lead to sub-optimal models (e.g. Weiss [2004], Kubat and Matwin [1997]) and misleading conclusions (e.g. Ranawana and Palade [2006], Ribeiro [2011]). As in utility-based learning, special purpose metrics are required for imbalanced domains learning problems. However, choosing such metrics is not as straightforward as it is in utility-based learning. In utility-based problems, the goal is the maximisation of the utility, and therefore, utility-based metrics (cf. Equations 2.19) are the obvious choice. However, in the context of imbalanced domains the information on the user preferences is typically less formal and expressed as an interest on the performance in "*the rare cases*", instead of a full utility matrix or surface. Therefore, the issue of performance assessment is hard when tackling imbalanced domains due to the difficulty in capturing with precision the user preferences. Still, multiple solutions have been proposed for assessing the performance in imbalanced domains. These solutions are mostly available for classification tasks, and in particular for binary class problems, which is also the type of task where more research exists.

We will present the performance assessment metrics categorised into scalar-based and graphical-based, as we have done for utility-based measures. Scalar metrics present the results as a single number while for graphical-based metrics the results are displayed or derived from graphical representations. Table 2.12 summarises the main references and metrics for performance assessment on imbalanced domains categorised by task and type of metric.

Table 2.12: Imbalanced domains performance assessment metrics and main bibliographic references for classification and regression tasks.

Metric type	Task type		Metric	Main References
Scalar	Classification	binary	$TP_{rate}(recall\ or\ sensitivity)$, $TN_{rate}(specificity)$, FP_{rate}, FN_{rate}, $PP_{value}(precision)$, NP_{value}, F_β, $G-Mean$, $dominance$, $IBA_\alpha(M)$, CWA, $balanced\ accuracy$, $optimized\ precision$, $adjusted\ G-Mean$, B_{42}	Rijsbergen [1979], Kubat et al. [1998], Estabrooks and Japkowicz [2001], Cohen et al. [2006], Ranawana and Palade [2006], García et al. [2008, 2009], Batuwita and Palade [2009], Brodersen et al. [2010], García et al. [2010], Thai-Nghe et al. [2011], Batuwita and Palade [2012]
		multiclass	$recall(c)$, $precision(c)$, $F_\beta(c)$, Rec_μ, $Prec_\mu$, Rec_M, $Prec_M$, MF_β, $MF_{\beta\mu}$, $MF_{\beta M}$, $MAvA$, $MAvG$, CWA	Sun et al. [2006], Ferri et al. [2009], Sokolova and Lapalme [2009]
	Regression		NMU, $precision^u$, $recall^u$, $precision^\phi$, $recall^\phi$	Torgo and Ribeiro [2007, 2009], Ribeiro [2011], Branco [2014]
Graphical	Classification	binary	$ROC\ curve$, AUC, $ProbAUC$, $ScoredAUC$, $WAUC$, $PR\ curve$, $Cost\ curve$, $Brier\ curve$,	Egan [1975], Metz [1978], Bradley [1997], Provost and Fawcett [1997], Provost et al. [1998], Drummond and Holte [2000a], Ferri et al. [2005], Davis and Goadrich [2006], Fawcett [2006b], Wu et al. [2007], Weng and Poon [2008], Hand [2009], Ferri et al. [2011b,a]
		multiclass	$ROC\ surface$, $AUNU$, $AUNP$, $AU1U$, $AU1P$, $SAUC$, $PAUC$	Mossman [1999], Ferri et al. [2009], Alejo et al. [2013], Sánchez-Crisostomo et al. [2014]
	Regression		$AUC-ROC^\phi$, $AUC-PR^\phi$, $AUC-ROCIV^\phi$, $AUC-PRIV^\phi$, $REC\ surface$	Torgo [2005], Ribeiro [2011]

It is also important to highlight that the conclusions drawn from the use of different evaluation metrics may be different (e.g. Van Hulse et al. [2007]). This may be a problematic issue and reinforces the need to develop suitable metrics that are able to correctly adapt to the user preferences.

2.4.2.1 Metrics for Imbalanced Classification Tasks

Let us consider a test set with n cases each belonging to one class $c \in \mathbb{C}$. For each test case $\langle \mathbf{x}_i, y_i \rangle$, let $\hat{y}_i = h(\mathbf{x}_i)$ represent the class predicted by model h. The estimated conditional probability of example $\mathbf{x}_i$ belonging to class c is represented by $\hat{p}(Y = c \,|\, X = \mathbf{x}_i)$, or $\hat{p}(c \,|\, \mathbf{x}_i)$. An indicator function that returns 1 when its argument is true and 0 otherwise will be represented by $I()$. The total number of examples belonging to a class c is represented

by $n_c = \sum_{i=1}^{n} I(y_i = c)$. The prior probability of a given class c can be estimated as $p(Y = c) = \frac{n_c}{n}$. For the special case of binary classification task, we consider the two classes $y \in \{0, 1\}$, where 0 will represent the negative (more frequent or majority) class and 1 will represent the positive (less frequent or minority) class.

Scalar Metrics

The confusion matrix for a two-class problem summarises the results obtained by a given classifier in a test set (cf. Table 2.13). This table provides for each class the cases that were correctly classified, i.e. the number of True Positives (TP) and True Negatives (TN), and the cases that were misclassified, i.e. the number of False Positives (FP) and False Negatives (FN).

Table 2.13: Confusion matrix for a two-class problem.

		True		**Total**
		Negative ($y = 0$)	Positive ($y = 1$)	
Predicted	Negative ($y = 0$)	TN= $\sum_{i=1}^{n} I(y_i = 0)I(\hat{y}_i = 0)$	FN $= n_+ - TP$	$\sum_{i=1}^{n} I(\hat{y}_i = 0)$
	Positive ($y = 1$)	FP $= n_- - TN$	TP= $\sum_{i=1}^{n} I(y_i = 1)I(\hat{y}_i = 1)$	$\sum_{i=1}^{n} I(\hat{y}_i = 1)$
Total		$n_- = \sum_{i=1}^{n} I(y_i = 0)$	$n_+ = \sum_{i=1}^{n} I(y_i = 1)$	n

Several proposals were made to condense the information on the confusion matrix while taking into account the user preferences. However, given that frequently only strictly informal information is available, the data distribution is usually taken into account. The motivation behind this is related with the existence of a bias towards the less frequent classes. Using Table 2.13 the following measures (cf. Equations 2.21-2.26) can be derived,

$$true\ positive\ rate\ (recall\ or\ sensitivity): \ TP_{rate} = \frac{TP}{TP+FN} \tag{2.21}$$

$$true\ negative\ rate\ (specificity\ ): \ TN_{rate} = \frac{TN}{TN+FP} \tag{2.22}$$

$$false\ positive\ rate: \ FP_{rate} = \frac{FP}{TN+FP} \tag{2.23}$$

$$false\ negative\ rate: \ FN_{rate} = \frac{FN}{TP+FN} \tag{2.24}$$

$$positive\ predictive\ value\ (precision\ ):\ PP_{value} = \frac{TP}{TP+FP} \tag{2.25}$$

$$negative\ predictive\ value:\ NP_{value} = \frac{TN}{TN+FN} \tag{2.26}$$

Each of the described metrics aggregates the information under a different evaluation perspective. Thus, it is necessary to monitor several metrics simultaneously, which is impractical and difficult when evaluating the models performance. To overcome this problem, several composite measures were proposed for assessing the performance in imbalanced domains, such as, the F_β [Rijsbergen, 1979] or the *Geometric-Mean* (G-Mean) [Kubat et al., 1998].

The F_β metric (cf. Equation 2.27) integrates both *precision* and *recall* measures. The β parameter is used for setting the relative importance of *recall* with respect to *precision*. Commonly, the value of β is set to 1, which means that the same importance is given to *precision* and *recall*, originating the F_1 metric. Values of β larger than 1 increase the weight of *recall* whilst values smaller than 1 give more importance to *precision*.

$$F_\beta = \frac{(1+\beta)^2 \cdot recall \cdot precision}{\beta^2 \cdot precision + recall} \tag{2.27}$$

The F_β measure is popular in the context of imbalanced domains, providing a more reliable information regarding the models effectiveness on the important cases for the user, as opposed to standard measures, such as accuracy (e.g. Estabrooks and Japkowicz [2001]).

The results of G-Mean (cf. Equation 2.28) are also frequently reported. This measure represents the geometric mean of the classes accuracy. However, the same importance is given to both classes in this formulation. This motivated a new version of G-Mean where *specificity* is replaced by *precision*, in an attempt to focus the evaluation on the positive class.

$$G\text{-}Mean = \sqrt{\frac{TP}{TP+FN} \times \frac{TN}{TN+FP}} = \sqrt{sensitivity \times specificity} \tag{2.28}$$

García et al. [2008] proposed the use of *dominance* (cf. Equation 2.29) to address the inability of G-Mean to explain the contribution of each class to the overall performance. This measure takes values in $[-1, 1]$, where the extremes of the interval represent situations where a perfect accuracy is achieved in one class and all the cases of the other class where mislabelled. Achieving a *dominance* of $+1$ means that all minority class cases where correctly predicted and all negative class cases where misclassified. The opposite occurs when the *dominance* is -1.

$$dominance = TP_{rate} - TN_{rate} \tag{2.29}$$

The *Mean Class Weighted Accuracy* (CWA), was proposed by Cohen et al. [2006] to address limitations of both F_β and G-Mean. This measure tries, simultaneously, to address the drawback of F_β, that does not accounts for the performance on the negative class, and the limitation of G-Mean, that does not assign more importance to the minority class. The CWA measure (cf. Equation 2.30) solves these issues transferring the responsibility of setting the weights used (parameter $w \in [0, 1]$) to the user.

$$CWA = w \cdot sensitivity + (1 - w) \cdot specificity \tag{2.30}$$

Several other metrics, such as *optimized precision* [Ranawana and Palade, 2006], *adjusted geometric mean* [Batuwita and Palade, 2009, 2012] or *B42* [Thai-Nghe et al., 2011], were proposed with similar objectives.

The performance assessment metrics described above are suitable for binary classification problems. Multiple proposals have been put forward for allowing their extension to a multiclass setting. This extension can be accomplished in several ways, which makes the problem of performance assessment more difficult in this context. For instance, *precision* and *recall* were extended to multiclass (cf. Equation 2.31 to 2.35) using two strategies: micro (μ) and macro (M) averaging. The macro averaging strategy averages the metric results over all classes while the micro averaging strategy uses the pooled results. In imbalanced domains the macro averaging strategy assigns the same weight to the classes while in the micro averaging strategy the more frequent classes have more importance. Therefore, metrics derived using micro averaging are usually considered unsuitable for performance assessment in imbalanced domains. We must highlight that, for the micro averaging strategy both Rec_μ and $Prec_\mu$ produce the same result. This happens because the numerator is equal in both equations and in Rec_μ the denominator is the sum of all true cases, while for the $Prec_\mu$ the denominator sums all the predicted cases for all cases. In both cases, this sum is the total number of examples, and therefore, Rec_μ and $Prec_\mu$ are the same.

$$Rec_\mu = \frac{\sum_{i=1}^{n} I(y_i = \hat{y}_i)}{\sum_{i=1}^{n} I(y_i = i)} \tag{2.31}$$

$$Prec_\mu = \frac{\sum_{i=1}^{n} I(y_i = \hat{y}_i)}{\sum_{i=1}^{n} I(\hat{y}_i = i)} \tag{2.32}$$

$$Rec_\mu = Prec_\mu = \frac{\sum_{i=1}^{n} I(y_i = \hat{y}_i)}{n} \tag{2.33}$$

$$Rec_M = \frac{\sum_{c \in \mathbb{C}} recall(c)}{|\mathbb{C}|} \tag{2.34}$$

$$Prec_M = \frac{\sum_{c \in \mathbb{C}} precision(c)}{|\mathbb{C}|} \tag{2.35}$$

Ferri et al. [2009] proposed the extension of F_β to multiclass as follows,

$$MF_\beta = \frac{\sum_{c \in \mathbb{C}} F_\beta(c)}{|\mathbb{C}|} \tag{2.36}$$

However, Sokolova and Lapalme [2009] also proposed two alternative ways for obtaining F_β in multiclass problems. These metrics use either the micro or macro averaging strategies to evaluate precision and recall which are then used in the F_β standard formulation (cf. Equation 2.27).

This demonstrates that the conversion of a measure to a multiclass scenario is not simple. The generalisation to multiclass of other metrics was also proposed (e.g. Sun et al. [2006], Cohen et al. [2006]).

Graphical-based Metrics

ROC curves (cf. Figure 2.6) and the Area Under the ROC Curve (AUC-ROC) measure [Metz, 1978] were proposed as alternatives to the use of standard evaluation measures in imbalanced domains by Provost et al. [1998]. The ROC curve allows to visualise the trade-off between TP_{rate} and FP_{rate}. A ROC curve is composed by a set of points each one corresponding to the use of a different decision threshold for classifying an example as positive. However, the use of ROC curves may be impractical for comparing the performance of different models because it may not be straightforward which models is the best unless one curve dominates all the others [Provost and Fawcett, 1997]. In Figure 2.6 we can observe that classifier C presents the worst performance. However, it is not easy to decide if the best model is A or B when we only take into account the ROC curve information. The AUC-ROC aggregates under a single value the performance of the models, making easier the task of deciding which one is the best on average. The AUC-ROC is given by a definite integral. The approximation of this integral provided by the trapezoidal method is one of the most widely used in practice. This method estimates the AUC-ROC value using trapezoids built with linear interpolation of points in the ROC curve. In the situation depicted in Figure 2.6, the AUC-ROC of model A is higher, and therefore, this model is preferred when considering this metric. Fawcett [2006b] showed the equivalence between AUC-ROC and the Wilcoxon test of ranks. Using this property, AUC-ROC can be determined as follows:

$$AUC(c, c') = \frac{\sum_{i=1}^{n} I(y_i = c) \sum_{t=1}^{n} I(y_t = c') L(\hat{p}(c \,|\, x_i), \hat{p}(c \,|\, x_t))}{n_c \cdot n_{c'}} \tag{2.37}$$

where c and c' are the two classes of the problem and L is a function that returns 0.5 when the two arguments are equal, returns 1 when the first argument is greater than the second and returns 0 otherwise.

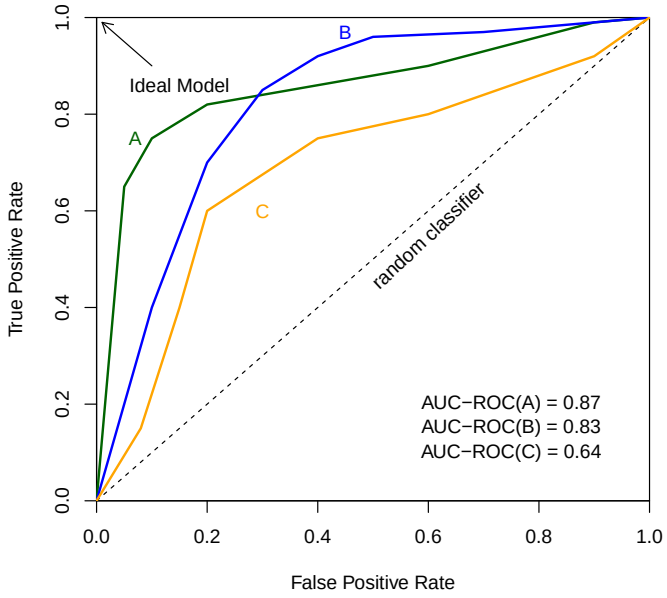


Figure 2.6: ROC curves of 4 classifiers (A, B, C and Random) and corresponding AUC-ROC.

Still, we must highlight that AUC-ROC measure is not biased towards the minority class. Moreover, the coherence of AUC-ROC was questioned by Hand [2009], that detected variations in the evaluation of AUC-ROC for different classifiers. Therefore, AUC-ROC can be misleading as it evaluates different classifiers using different measures. Still, Ferri et al. [2011b] also presented a possible coherent interpretation for this measure.

Several adaptations of AUC-ROC were proposed, such as the *ProbAUC* [Ferri et al., 2005], the *ScoredAUC* [Wu et al., 2007] or the *WeightAUC* [Weng and Poon, 2008]. *ProbAUC* tries to overcome the limitation of AUC-ROC of not accounting for the probabilities of the examples. *ScoredAUC* consider these probabilities but tries to provide a score more robust to small changes in the probabilities. Finally, *WeightAUC* assigns more importance to the areas near the top of the ROC curve, that are considered by the authors to be more relevant in imbalanced domains.

The inability of ROC curves to deal with problems in which the error costs vary with the instances lead to the proposal of Instance Varying Receiver Operating Characteristics curve (ROCIV) by Fawcett [2006a]. The Area Under the ROCIV Curve (AUC-ROCIV) is obtained in a similar way to AUC-ROC with the difference that the instances are chosen in proportion to their costs.

Precision-Recall Curves (PR Curves) are another important graphical tool in the context of imbalanced domains. These curves present the tradeoff between precision and recall rates

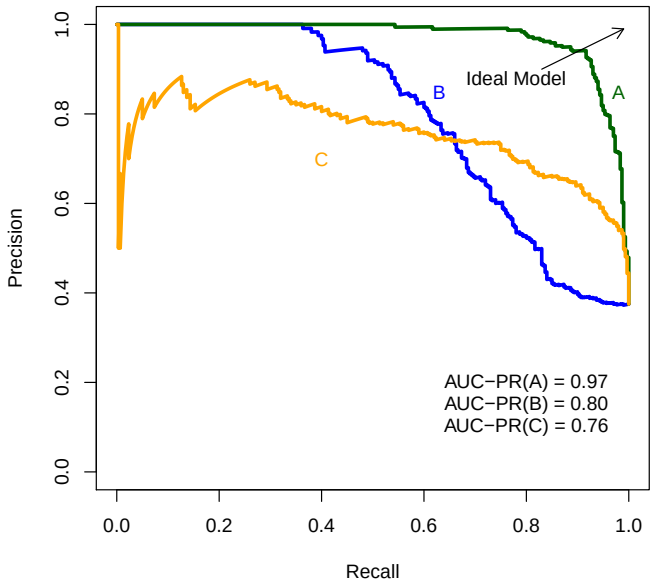


Figure 2.7: PR Curve curves of 3 classifiers and corresponding AUC-PR.

and are recommended for highly imbalanced domains where ROC curves may provide an optimistic view of the performance [Davis and Goadrich, 2006]. The Area Under the PR Curve (AUC-PR) was also proposed for summarising the PR Curve information [Davis and Goadrich, 2006]. Figure 2.7 displays three PR Curve curves with the corresponding AUC-PR measure. As we can observe, the ideal model in PR Curve curves corresponds to the top right corner, where both precision and recall achieve the maximum value.

Addressing the issue of performance evaluation in the multiclass scenario using graphical-based metrics is a complex task. For instance, for obtaining ROC curves in a problem with c classes it is possible to use the strategy one-vs-all. When using this strategy, each class is considered the positive class at a time and the remaining classes are aggregated into a negative class, generating $|\mathbb{C}|$ different two-class problems. Mossman [1999] proposed the *ROC surface* for the specific case of classification problems with three classes. However, when the number of classes increases the complexity of constructing these curves also grows. Another possibility is to consider all the combinations of pairs of classes, which increases more the complexity of the task because, in this case, $|\mathbb{C}|(|\mathbb{C}| - 1)$ two-class problems are obtained.

Several adaptations of AUC-ROC to the multiclass setting were proposed [Ferri et al., 2009]. These proposals vary in the assumptions they make and the strategy followed to decompose multiclass into binary class problems. Some adaptations assume that the classes are uniformly distributed while others take into account the prior probability of each class.

Regarding the strategy for decomposing the multiclass problem into several binary class problems, some adaptations use the one-vs-all strategy while others evaluate all pairs of classes.

The existing solutions for multiclass imbalanced problems are still scarce. Moreover, some recent works [Alejo et al., 2013, Sánchez-Crisostomo et al., 2014] have shown that these metrics may not be reliable for assessing the performance in imbalanced domains because they do not always reflect the performance on the minority and majority classes.

2.4.2.2 Metrics for Imbalanced Regression Tasks

In the context of imbalanced regression tasks, few efforts have been made regarding the proposal of suitable evaluation metrics. Ribeiro [2011] showed that standard regression measures, such as Mean Squared Error (MSE) or Mean Absolute Error (MAE), are inadequate in this context. Ideally, the performance evaluation in imbalanced regression tasks should take into account both the magnitude of the error, and the location of the error within the target variable domain. This means that the error magnitude should have a different impact in the performance according to the user preferences over the target variable domain.

Scalar Metrics

Ribeiro [2011] has presented an approach based on the utility-based regression framework described in Torgo and Ribeiro [2007]. This proposal allows to automatically specify a utility surface for applications where the goal is to be accurate on rare extreme values of a continuous target variable. These applications are a specific type of imbalanced regression where rarity occurs in the extreme values of the domain. In spite of this specificity, this is actually a rather frequent setting in real world applications (e.g. ecological modelling, financial forecasting, etc.). We will describe this framework in more detail in Chapter 3. Using this method, Ribeiro [2011] proposed the Normalised Mean Utility (NMU), which is a normalised version of MU measure that uses the utility framework described. This metric can be used when the framework assumptions are fulfilled.

The utility-based regression framework proposed by Ribeiro [2011] proposes the introduction of the auxiliary notion of relevance function. This function, named $\phi(Y)$, expresses the domain-specific bias regarding the target variable domain $\mathcal{Y}$. The relevance function maps the target variable values into a relevance (importance) scale, where 1 corresponds to the maximal relevance and 0 to minimum relevance.

To alleviate the end-user effort when defining the relevance function, Ribeiro [2011] proposed an automatic method for estimating $\phi(Y)$. This method assumes that the relevance is inversely proportional to the target variable probability density function. In this setting, the

most interesting ranges of the domain are associated with higher levels of rarity. Moreover, the rare and interesting values of the target variable are concentrated on the extremes of the distribution. The proposed automatic method is only suitable for the particular sub-class of imbalanced regression where the more relevant cases are the rare extreme values. The estimate of $\phi(Y)$ is obtained by using the estimated quartiles and the inter-quartile range of the target variable distribution.

Based on the described utility framework , Branco [2014] proposed $prec^{\phi}$ and rec^{ϕ} metrics, which adapt the concepts of *precision* and *recall* defined for classification tasks to an imbalanced regression setting,

$$prec^{\phi} = \frac{\sum_{\phi(\hat{y}_i)>t_R}(1+u(\hat{y}_i,y_i))}{\sum_{\phi(\hat{y}_i)>t_R}(1+\phi(\hat{y}_i))} \tag{2.38}$$

$$rec^{\phi} = \frac{\sum_{\phi(y_i)>t_R}(1+u(\hat{y}_i,y_i))}{\sum_{\phi(y_i)>t_R}(1+\phi(y_i))} \tag{2.39}$$

where $\phi()$ is the relevance function, y_i, and $\hat{y}_i$ are the true and predicted values of a case, t_R is a user-defined threshold signalling the cases that are relevant for the user, and $u(\hat{y}_i, y_i)$ is the utility of making the prediction $\hat{y}_i$ for the true value y_i, normalised to $[-1, 1]$.

Graphical-based Metrics

Ribeiro [2011] proposed several measures integrated in the utility framework suitable for forecasting rare extreme values. In this context, the AUC-ROC and AUC-PR measures were adapted to regression problems ($AUC-ROC^{\phi}$ and $AUC-PR^{\phi}$ respectively) by Ribeiro [2011]. These adaptations followed the same definitions provided for classification but replaced the base classification metrics results by the corresponding in regression. To overcome the inability of $AUC-ROC^{\phi}$ to account for the different utility of the instances, Ribeiro [2011] also proposed the Area Under the ROCIV Curve for regression ($AUC-ROCIV^{\phi}$), an adaptation of AUC-ROCIV notion to regression. This was accomplished through the utility framework assuming the selection of instances proportionally to their utility. Following the same intuition a similar adaptation was proposed to derive the Area under the Instance Varying Precision-Recall Curve for regression ($AUC-PRIV^{\phi}$). Still, we must highlight that these measures assume a utility framework developed specifically for a particular class of imbalanced regression problems where the rare and relevant values are located at the extremes of the target variable distribution

REC Curves were presented in Section 2.3.2 (page 26) in the context of graphical-based utility metrics. Still, in the context of imbalanced domains, REC Curves have an important drawback: they are insensitive to the error location across the target variable domain. To

solve this problem, Torgo [2005] proposed the use of Regression Error Characteristic Surfaces (REC Surfaces). These surfaces add an extra dimension to REC Curves showing the errors distribution across the target variable range. This means that one point in the REC Curve is mapped into a curve in the REC Surface that displays the distribution of the errors over the target variable range. Figure 2.8 shows an example of a REC Surface. REC Surface are useful in the context of imbalanced domains because they allow: i) the observation of the error distribution across the problem domain; and ii) the inspection of the types of errors specifically in the relevant ranges of the target variable.

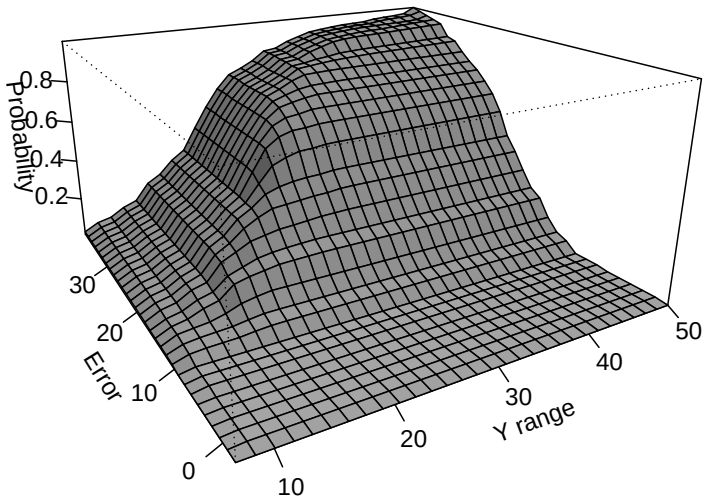


Figure 2.8: An example of a REC Surface.

2.4.3 Methods for Dealing with Imbalanced Domains

When dealing with imbalanced domains it is necessary to cope with the learning algorithms inability to focus on the important cases. This corresponds to one of the challenges shared with utility-based tasks described in Section 2.2. Still, in the case of imbalanced domains this challenge is even harder given the rarity of the important cases. Several strategies were developed to face this problem. Still, the majority of solutions proposed are suitable for classification tasks, and in particular for binary classification. So far, few solutions are suitable for the multiclass or regression contexts.

Similarly to the utility-based learning taxonomy, the approaches for tackling the problem of imbalanced domains can be grouped into direct, meta-learning and hybrid methods. Figure 2.9 shows the taxonomy of strategies for dealing with imbalanced domains as proposed by Branco et al. [2016b].

As we previously described, direct methods change the learning algorithm for improving its focus on the most relevant cases. Direct methods can be further categorised into utility-based methods and other methods not dependent on the utility information. Meta-learning

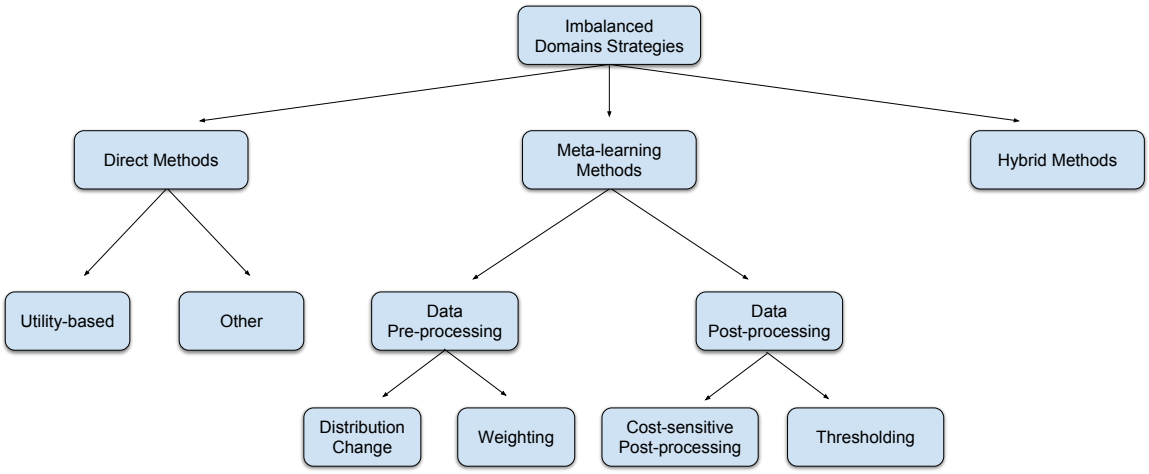


Figure 2.9: Taxonomy of strategies for tackling imbalanced domains problems [Branco et al., 2016b].

methods use standard learning algorithms and act before or after them. These methods can be further categorised into: data pre-processing (change the training set) and data post-processing (change the predictions) methods. Finally, hybrid strategies include methods that apply changes in several moments of the learning procedure. The main advantages and disadvantages of the existing methods are summarised in Table 2.14. In the following sections we present an overview of the existing proposals for the types of strategies described.

2.4.3.1 Direct Methods

The direct methods for dealing with imbalanced domains aim at modifying or developing new learning algorithms for providing a better fit to the domain specific bias. This is a challenging task because it requires both the deep knowledge of the algorithms and the capability of mapping the user information into an effective change. To modify a learner, it is necessary to understand: i) why it fails when applied to the given domain; and ii) which modifications are more adequate to match the expressed (formally or informally) user preferences. When the available information is provided in a very informal way, this task becomes extremely difficult. Nevertheless, when the changes in the learning algorithm are accomplished, this becomes a very effective solution for the context for which it was developed. The main bibliographic references regarding direct methods are summarised in Table 2.15.

One way of changing the learning algorithms to deal with imbalanced domains is through the incorporation of the utility information. This includes all cost-sensitive and, more generally, utility-based algorithms for classification and regression tasks that directly include the cost/benefit information into the learning process.

Table 2.14: Main advantages and disadvantages of each type of strategy for imbalanced domains learning.

Strategy	Advantages	Disadvantages
Direct Methods	• user goals are incorporated directly into the models • models obtained are more comprehensible to the user	• user is restricted to learning algorithms that have been modified to be able to optimize his goals • models must be relearned if the target loss function changes • changes in the loss function may require further modifications in the algorithm • requires a deep knowledge of the learning algorithms implementations • not easy to map the specified user preferences into a loss function
Meta-learning Methods: Pre-processing	• can be applied to any learning tool • the chosen models are biased to the goals of the user • models more interpretable according to the user goals	• difficulty of relating the modifications in the data distribution and the user preferences
Meta-learning Methods: Post-processing	• it is not necessary to be aware of the user preference bias at learning time • the obtained model can, in the future, be applied to different deployment scenarios without the need of re-learning the models or even keeping the training data available • any standard learning tool can be used	• the models do not reflect the user preferences • models interpretability may be jeopardised as they were obtained optimising a loss function that is not in accordance with the user preference bias

Table 2.15: Main bibliographic references on direct methods for imbalanced domains chronologically ordered

Strategy type		Main References
Direct Methods	**Utility-based**	Joshi et al. [2001], Maloof [2003], Akbani et al. [2004], Chen et al. [2004], Zhou and Liu [2006], Alejo et al. [2007], Sun et al. [2007], Song et al. [2009], Wang and Japkowicz [2010], Oh [2011], Ribeiro [2011], Weiguo et al. [2012], Cao et al. [2013], Castro and de Pádua Braga [2013]
	Other	Barandela et al. [2003], Ribeiro and Torgo [2003], Tan et al. [2003], Torgo and Ribeiro [2003], Wu and Chang [2003b], Huang et al. [2004], Wu and Chang [2005], Imam et al. [2006], Tang and Zhang [2006], Cieslak and Chawla [2008], Li et al. [2009], Tang et al. [2009], Batuwita and Palade [2010b], Liu et al. [2010], Hwang et al. [2011], Cieslak et al. [2012], Rodríguez et al. [2012], Xiao et al. [2012]

Multiple works address the problem of converting standard classifiers into cost-sensitive ones. These adaptations have been already discussed in Section 2.3.3.1 of this thesis. However, some of these algorithms were adapted specifically to incorporate costs/benefits in the framework of imbalanced domains. This is what distinguishes them from the previously described. Typically they must deal with informal information regarding the cost/benefit values, and their main strategy involves developing utility incorporation mechanisms for facing the difficulties posed by imbalanced domains to standard learners. We will begin by describing these methods and then describe other methods that are not based on the utility setting.

Maloof [2003] addressed the problem imbalanced domains with regression trees, by incorporating costs that are both unknown and unequal. For SVMs several strategies were considered for embedding costs such as introducing weights in the attributes [Yuanhong et al., 2009], assigning a different penalisation to errors [Akbani et al., 2004], among other [Weiguo et al., 2012]. Costs were also included in NNETs with the exploration of several strategies for adapting them (e.g. Zhou and Liu [2006], Oh [2011]). Different strategies were proposed in this context, including modifications on the weight update rule [Castro and de Pádua Braga, 2013], the incorporation of Particle Swarm Optimization (PSO) [Cao et al., 2013] or embedding a cost function in the training phase [Alejo et al., 2007]. Multiple ensemble models have also been adjusted to incorporate costs/benefits with the goal of solving the problem of imbalanced domains. Boosting has been the most thoroughly explored, and in particular, the AdaBoost algorithm. The existing proposals modify the weight updating process in different ways. Examples of such proposals are the RareBoost [Joshi et al., 2001], AdaC1, AdaC2 and AdaC3 [Sun et al., 2007] and BABoost [Song et al., 2009]. A system using boosting and SVMs modified to incorporate costs was proposed by Wang and Japkowicz [2010]. Random

forest algorithm was also modified to deal with imbalanced domains, in a new Weighted Random Forest (WRF) algorithm [Chen et al., 2004] that assigns higher misclassification costs to the minority class.

Direct methods also integrate other strategies that modify the algorithms' preference criteria and are not bound to the utility concept. Modifications were proposed for SVMs that change the kernel function [Wu and Chang, 2003b], change SVM algorithms developed for dealing with outliers and noisy examples [Batuwita and Palade, 2010b], or define a new objective function and constraints [Li et al., 2009]. For k-Nearest Neighbours (k-NN), a new weighted distance function was integrated [Barandela et al., 2003]. The modifications proposed in decision trees for improving their performance when handling imbalanced domains are mostly focused on changes in the splitting criteria. For instance, the Class Confidence Proportion Decision Tree (CCPDT) proposed by Liu et al. [2010] adopts a new measure that is embedded in the information gain and used as the splitting criteria. The Hellinger distance was also incorporated as a splitting criteria in Hellinger Distance Decision Trees (HDDT) [Cieslak and Chawla, 2008] and bagged HDDTs [Cieslak et al., 2012] were recommended for learning in imbalanced domains. Modification to the splitting criterion of regression trees were also proposed by Torgo and Ribeiro [2003] and Ribeiro and Torgo [2003] in the context of rare cases prediction.

2.4.3.2 Pre-processing Meta-learning Methods

Pre-processing methods tackle the problem of learning from imbalanced domains by changing the distribution of the available training data before applying the learning algorithm. The goal is to obtain a new modified training set that is more in accordance with the user preferences. These modified training sets do not have the proobles of imabalanced data sets and thus allow the usage of standard learners without any modification. This is a big advantage that explains the popularity of these methods. However, it is also difficulty to determine the relationship between the modifications that are necessary to apply and the user preferences. This task becomes even more challenging when the information provided by the user is too informal.

Pre-processing methods can be further categorised into: methods that change the data distribution and methods that weight the data space. The former change the data distribution for addressing the issue of the lack of representativeness of the most important cases, while the latter use the cost/benefit information to modify the data distribution. Table 2.16 summarises the main sub-types of pre-processing methods and the corresponding bibliographic references.

Distribution Change. The key idea of these methods is to change the data distribution by somehow manipulating the available examples. The modification is achieved by including

Table 2.16: Main bibliographic references of pre-processing methods for imbalanced domains chronologically ordered.

Strategy type			Main References
Distribution Change	Stratified Sampling	Random Under/Over-sampling	Kubat and Matwin [1997], Japkowicz [2000b], Chawla et al. [2002], Chang et al. [2003], Drummond and Holte [2003], Chen et al. [2004], Estabrooks et al. [2004], Tao et al. [2006], Wang and Yao [2009], Seiffert et al. [2010], Wallace et al. [2011], Torgo et al. [2013]
		Distance Based	Chyi [2003], Mani and Zhang [2003], Błaszczyński and Stefanowski [2015]
		Data Cleaning Based	Kubat and Matwin [1997], Laurikkala [2001], Batista et al. [2004], Naganjaneyulu and Kuppa [2013]
		Recognition Based	Japkowicz [2000a], Chawla et al. [2004], Raskutti and Kowalczyk [2004], Lee and Cho [2006], Zhuang and Dai [2006b,a], Bellinger et al. [2012], Wagstaff et al. [2013]
		Cluster Based	Jo and Japkowicz [2004], Cohen et al. [2006], Yen and Lee [2006, 2009], Sobhani et al. [2014], Ofek et al. [2017]
		Evolutionary Sampling	Del Castillo and Serrano [2004], García et al. [2006], Doucette and Heywood [2008], Drown et al. [2009], García and Herrera [2009], Maheshwari et al. [2011], García et al. [2012], Yong [2012], Galar et al. [2013]
	Synthesising New Data		Lee [1999, 2000], Chawla et al. [2002, 2003], Batista et al. [2004], Han et al. [2005], Liu et al. [2007], He et al. [2008], Bunkhumpornpat et al. [2009], Hu et al. [2009], Wang and Yao [2009], Menardi and Torelli [2010], Maciejewski and Stefanowski [2011], Zhang et al. [2011], Barua et al. [2012], Bunkhumpornpat et al. [2012], Martínez-García et al. [2012], Ramentol et al. [2012a,b], Verbiest et al. [2012], Nakamura et al. [2013], Torgo et al. [2013], Gao et al. [2014], Li et al. [2014], Zhang and Li [2014], Bellinger et al. [2015], Sáez et al. [2015], Bellinger et al. [2018]
	Combination of Methods		Liu et al. [2006], Mease et al. [2007], Li et al. [2008], Stefanowski and Wilk [2008], Chen et al. [2010], Jeatrakul et al. [2010], Napierała et al. [2010], Songwattanasiri and Sinapiromsaran [2010], Bunkhumpornpat et al. [2011], Vasu and Ravi [2011], Sharma et al. [2012], Yang and Gao [2012], Ng et al. [2014]
Weighting the Data Space			Zadrozny et al. [2003]

or removing examples using some strategy.

The major obstacle of these methods is related with the difficulty in relating the user preference bias with the modifications to apply on the data. Even in situations where the user is able to fully specify the utility information, to decide which is the best data distribution to use is not straightforward. A solution that has shown to be effective to deal with imbalanced domains is to balance the distribution of the important and unimportant cases (e.g. Estabrooks et al. [2004], Fernández et al. [2008, 2010], Batuwita and Palade [2010a]). Nevertheless, it was also shown that, for some classifiers such as C4.5, Ripper or Naive Bayes, the optimal results are not always achieved with a perfectly balanced distribution [Weiss and Provost, 2003]. This motivated several works that search for the right amount of change to apply in the data distribution [Weiss and Provost, 2003, Chawla et al., 2005, 2008, Khoshgoftaar et al., 2007].

For the particular case of binary imbalanced classification problems, Breiman et al. [1984] showed the equivalence between the concepts of changing the data distribution and the misclassification cost ratio. However, as reported by Weiss [2013], this equivalence does not hold in many real world problems because of its assumptions on data availability.

The approaches for changing the data distribution may be categorised into: stratified sampling, synthesising new data or a combination of both. Stratified sampling methods add or remove examples from the original data set by considering a given technique that may be simply random, or based in distance measures, evolutionary algorithms, data cleaning, clustering or recognition approaches.

The simplest approaches involve the random selection of examples to include or to remove from the data set, i.e., random under-sampling and random over-sampling. The former randomly removes examples from the most frequent and uninteresting class while the latter adds replicas of examples belonging to the less frequent and most important class. Random under-sampling strategy has as consequence the reduction of the data set size which may be an advantage. On the other hand, it discards examples that may be useful and may lead to a performance loss. The application of random over-sampling may increase the probability of overfitting, specially for higher rates of over-sampling [Chawla et al., 2002, Drummond and Holte, 2003]. Therefore, it may also negatively impact the performance and it increases the computational effort due to the larger training set. The random under-sampling strategy was also extended to regression tasks [Torgo et al., 2013]. To achieve this the user should provide a relevance function specifying the importance of the target variable domain and a threshold on the relevance scores that determines which are the important and unimportant cases. Random under-sampling was also embedded in ensemble methods such as boosting [Seiffert et al., 2010], bagging [Chang et al., 2003, Tao et al., 2006, Wang and Yao, 2009, Wallace et al., 2011] and random forest [Chen et al., 2004].

The modifications in the data distribution can also be achieved through other strategies

that change the data in a more informed and non-random way. This happens with methods based on distance between cases (e.g. Chyi [2003], Mani and Zhang [2003]). These proposals apply under-sampling by using a certain distance criteria to determine which examples are included in the new data set.

Data cleaning techniques also allow to perform informed under-sampling, by identifying potential problematic cases. Examples of such strategies include the use of Tomek links [Batista et al., 2004], Condensed Nearest Neighbour Rule (CNN) [Hart, 1968] or One Sided Selection (OSS) Kubat and Matwin [1997].

Recognition-based methods perform the most extreme type of under-sampling: the entire minority class is discarded and only the majority class examples are used. In this case, the recognition-based methods such as one-class learners or autoencoders, try to build boundaries surrounding the majority class. Examples falling outside these boundaries are classified as being from the minority class. These methods are based on the similarity between a new case and the majority class. They require the definition of a threshold which will set when the similarity score is enough, or not, for the example to be classified as belonging to the majority class. Examples of recognition-based methods include one-class SVMs (e.g. Schölkopf et al. [2001], Manevitz and Yousef [2002], Raskutti and Kowalczyk [2004], Zhuang and Dai [2006a,b], Lee and Cho [2006]) and autoencoders (or autoassociators) (e.g. Japkowicz et al. [1995], Japkowicz [2000a]).

Another way to change the distribution of the original data set is by using clustering algorithms. An example of such methods is the Cluster-Based Over-sampling (CBO) strategy, proposed by Jo and Japkowicz [2004], where a clustering algorithm is used to deal with the problem of imbalanced domains and also to face the small disjuncts problem. Small disjuncts are subclusters of a class that contain few examples [Holte et al., 1989]. CBO method applies the well known k-means algorithm to each class. Then, each majority class cluster is over-sampled until it reaches the largest cluster size of this class. Finally, the minority class clusters are also over-sampled until the distribution is balanced. Several other approaches exist involving the use of clustering algorithms (e.g. Yen and Lee [2006], Cohen et al. [2006], Yen and Lee [2009]) and also integrating them with ensemble methods (e.g. Sobhani et al. [2014]).

Evolutionary Algorithms (EAs) have also been used as informed under- and over-sampling techniques to deal with imbalanced domains. For instance, García et al. [2006] and García and Herrera [2009] propose under-sampling methods through prototype selection procedures. EAs have also been used in conjunction with other techniques in the development of strategies that combine under- and over-sampling. For instance, the combination of genetic algorithms and clustering methods was proposed [Maheshwari et al., 2011, Yong, 2012], and EAs were also integrated with boosting methods [Galar et al., 2013].

The problem of imbalanced domains may also be addressed through the generation of new

synthetic cases. A diversity of proposals exist for synthesising new data due to its known advantages [Chawla et al., 2002, Menardi and Torelli, 2010], such as: i) the reduction of the overfitting probability when compared to the introduction of exact copies of examples; and ii) the improved generalisation capability. The methods that allow to obtain new synthetic examples can be categorised into those that only introduce small perturbations on original cases, and those that obtain new cases by an applying an interpolation strategy.

A strategy that generates new minority class examples by adding normally distributed noise to the minority class cases was proposed by Lee [1999] and Lee [2000]. The DEAGO system [Bellinger et al., 2015] is another example that also generates synthetic cases with Gaussian noise added.

The Random Over Sampling Examples (ROSE) framework [Menardi and Torelli, 2010] is based on a smoothed bootstrap re-sampling technique that combines over- and under-sampling to obtain a balanced and completely new training set. ROSE framework works as follows: i) an example is selected from the original data set by assigning the same probability to both classes; ii) a new synthetic case is generated in the neighbourhood of the case previously draw using a neighbourhood width determined by a user selected smoothing matrix.

One of the most successful strategies for obtaining new synthetic data is the Synthetic Minority Over-sampling TEchnique (SMOTE) algorithm proposed by Chawla et al. [2002]. SMOTE uses an interpolation strategy to obtain new synthetic examples from the minority class. The new synthetic case is built by using a minority class case and one of its k nearest neighbours randomly selected. The new case is generated randomly along the line segment joining the two cases as displayed in Figure 2.10. This technique is typically combined with random under-sampling that is applied to the majority class cases.

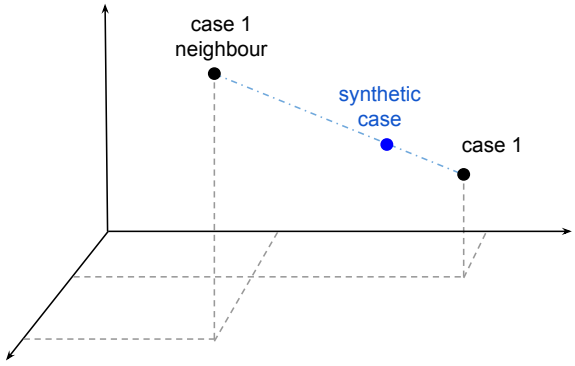


Figure 2.10: Example of synthetic case generation using SMOTE algorithm.

SMOTE has also been integrated with ensemble methods such as boosting [Chawla et al., 2003] and bagging [Wang and Yao, 2009].

Although SMOTE presents multiple advantages, specially when compared to over-sampling

strategies that simply introduce replicas of examples, several drawbacks have also been pointed to this strategy. For instance, the fact that the majority class cases are not taken into account when new cases are generated is one of the problems of SMOTE. Disregarding the majority class cases may lead to overgeneralization [Yen and Lee, 2006, 2009, Maciejewski and Stefanowski, 2011]. This may be particularly problematic for highly imbalanced domains where the minority class cases are very sparse and new synthetic cases can be generated with higher probability in overlapping regions of the problem domain. Several strategies were proposed to face these difficulties of the SMOTE algorithm. These variants can be aggregated into the following main types: i) apply a pre-/post-processing method before/after using SMOTE to avoid the generation of cases in inconvenient regions or remove cases generated in those regions; ii) apply SMOTE only in selected regions of the domain; iii) introduce modifications directly in the SMOTE algorithm.

The first SMOTE variant concerns essentially the use of some data cleaning technique before or after the application of SMOTE to remove cases that may hinder the classifiers performance. Some examples of the first type of SMOTE variants are SMOTE+Tomek [Batista et al., 2004], SMOTE+ENN [Batista et al., 2004], SMOTE+FRST [Ramentol et al., 2012b], SMOTE+RSB [Ramentol et al., 2012a] or Fuzzy Rough Imbalanced Prototype Selection (FRIPS) [Verbiest et al., 2012]. Regarding the second types of variants, there are several proposals that act in a very different way: some approaches try to reinforce the borders of the minority class, others reinforce only the domain space clearly belonging to the minority class, and other try to identify the cases that are difficulty to learn in order to reinforce only those cases. Some examples of this variant of SMOTE are: Borderline-SMOTE [Han et al., 2005], ADASYN [He et al., 2008], Modified Synthetic Minority Oversampling Technique (MSMOTE) [Hu et al., 2009], FSMOTE [Zhang et al., 2011], MWMOTE [Barua et al., 2012], among others. Finally the last type of SMOTE variants include changes to the SMOTE algorithm that bias the generation of synthetic cases towards some regions of the problem domain. Some proposals that fit into this category are: Safe-Level-SMOTE [Bunkhumpornpat et al., 2009], Safe Level Graph [Bunkhumpornpat and Subpaiboonkit, 2013], LN-SMOTE [Maciejewski and Stefanowski, 2011] and DBSMOTE [Bunkhumpornpat et al., 2012].

The SMOTE algorithm was also adapted to handle imbalanced regression tasks by Torgo et al. [2013]. In Chapter 5 we present an extended version of this algorithm that uses the same procedure for generating new cases, but is able to handle multiple important and unimportant regions of the target variable domain.

A large number of other methods have emerged that combine some of the previously described techniques (e.g. Stefanowski and Wilk [2008], Bunkhumpornpat et al. [2011], Songwattanasiri and Sinapiromsaran [2010], Yang and Gao [2012]). For instance, some proposals devote a special attention to biasing NNETs [Jeatrakul et al., 2010], SVMs [Kang and Cho, 2006, Liu et al., 2006], or ensemble methods [Mease et al., 2007, Chen et al., 2010, Galar

et al., 2012].

Weighting the Data Space. This can be an effective method to deal with imbalanced domains as it corresponds to implementing cost-sensitive learning. However, in this case, the information regarding the misclassification costs should be available. This method consists of applying misclassification costs to the original data set to determine the best modified training distribution. By assigning a weight to each case that is proportional to its importance, the original data distribution is changed.

This technique, although simple and easy to apply, has also some important drawbacks, such as, the increased probability of overfitting and the requirement of having cost information. This approach is theoretically substantiated by the *Translation Theorem* that was derived by Zadrozny et al. [2003]. Two different methods were proposed by Zadrozny et al. [2003] to change the training set taking into consideration the misclassification costs information. One method can be though as a transparent box, and consists of directly providing the weights to the classifier. The main disadvantage of this method is the need of a classifier that is able to handle case weights. The second method is a black box approach and consists of using the determined weights to sample the original training set. This method may lead to overfitting if sampling with replacement is used. To address the drawbacks of the black box approach, Zadrozny et al. [2003] proposed the cost-proportionate rejection sampling method. In this sampling procedure each example is included in the modified training set with a probability proportional to the respective weight.

2.4.3.3 Post-processing Meta-learning Methods

Post-processing methods act after the learning stage, changing the predictions outputted by a model learned using the original data set and a standard learning algorithm. The main solution for changing the predictions at a post-processing level is the thresholding method. This method uses a class membership score to derive a ranking of the examples and produces different learners by manipulating the class membership threshold. Table 2.17 summarises the main bibliographic references of post-processing meta-learning methods.

Table 2.17: Main bibliographic references of post-processing methods for imbalanced domains chronologically ordered.

Strategy type	Main References
Thresholding	Maloof [2003], Weiss [2004]

Proposals for handling imbalanced domains through the thresholding strategy assume that

Table 2.18: Main bibliographic references of hybrid strategies chronologically ordered.

Strategy type	Main References
Hybrid	Estabrooks and Japkowicz [2001], Kotsiantis and Pintelas [2003], Estabrooks et al. [2004], Phua et al. [2004], Yoon and Kwek [2005], Ertekin et al. [2007b,a], Zhu and Hovy [2007], Liu et al. [2009], Ghasemi et al. [2011a,b], Ertekin [2013], Mi [2013], Barnab-Lortie et al. [2015], Lim et al. [2017]

the used classifier is able to produce a score for each case that expresses the class membership degree of the example. These classifiers are frequently named soft classifiers. Typically, a decision threshold is used for determining the prediction for each case based on the class membership score. By varying the decision threshold, it is possible to obtain different predictions [Weiss, 2004]. Maloof [2003] studied the impact of moving the decision threshold, applying a sampling strategy and adjusting the cost matrix. This study provided empirical evidence that the classifiers obtained using the three methods have a similar performance.

2.4.3.4 Hybrid Methods

Hybrid methods combine previously developed methods, trying this way to minimise drawbacks detected in some approaches. Table 2.18 provides a summary of the main bibliographic references concerning hybrid methods.

Estabrooks and Japkowicz [2001] and Estabrooks et al. [2004] proposed one of the first hybrid methods. This method is a mixture-of-experts system that combines several sets of models learned in under-sampled and over-sampled training sets using different sampling rates. A specially developed strategy is applied to obtain the final predictions. Several systems were proposed that integrate the use of different learners and sampling strategies (e.g. Kotsiantis and Pintelas [2003], Phua et al. [2004], Liu et al. [2009], Wang [2008]).

Several other proposals merge the concept of active learning with meta-learning strategies (e.g. Ertekin et al. [2007b,a], Zhu and Hovy [2007], Ertekin [2013], Mi [2013]).

2.5 Conclusions

In this chapter we presented a thorough discussion of the related work regarding utility-based learning and the particular sub-problem of learning from imbalanced domains. We presented the existing solutions for dealing with the performance evaluation issue and we discussed the main learning methods proposed for tackling both problems.

Utility-based learning has been mainly addressed in a classification context. Several important theoretical results are available for binary classification tasks, while for the multiclass or regression settings these results are more scarce. Moreover, also regarding the development of new learning methods, there are only few available for multiclass problems and regression. Still, it is recognised that costs and benefits are inherent to a vast range of important real world applications. Therefore, it is necessary and important to address these open challenges.

The development of a general theoretical utility-based learning framework that incorporates both classification and regression tasks could provide important insights for this problem. However, as we have mentioned, a unifying theory for utility-based learning still does not exists.

One of the main obstacles when considering utility-based learning in regression tasks is related with obtaining the full utility information. In classification the user needs to provide a utility matrix, while for regression tasks a utility surface is required. To obtain this information in regression can be a demanding and time consuming task. It is important to develop new methods for tackling this issue.

Regarding the problem of imbalanced domains, one of the major challenges is related with the quality of the information that is available to the data analyst. If the full information regarding the user preferences is available, then the problem can be solved using the utility-based learning mechanisms. The challenges are larger when the information available is more informal. In this case, it is necessary to understand how to properly evaluate the performance and also which methods to apply to obtain better models according to the user preference bias. Another challenge of learning from imbalanced domains in regression tasks regards the lack of methods for tackling this particular class of problems.

Chapter 3

A Utility-based Learning Framework

3.1 Introduction

The main goal of this chapter is to provide a unifying framework for utility-based predictive analytics. This framework unifies how different learning tasks are understood and incorporates both classification and regression problems. Our goal is to formalise predictive tasks, generalising their main characteristics. This framework provides a broader overview of the learning problem showing important connections between different predictive tasks. We also discuss the relationship between utility-based learning and other predictive problems. Finally, we discuss the main open challenges of utility-based learning and imbalanced domains learning tasks. We focus our attention on the utility-based regression challenges because these are the most recently explored tasks and therefore, those with more unsolved issues.

3.2 Formalisation of Predictive Analytics

Predictive analytics has a key role in Data Science. Predictive analytics uses the available data and other relevant information from a given domain with the goal of building models that solve predictive tasks. The domain information that the end-user is capable of supplying can be a fundamental aspect when solving the predictive task. This domain knowledge may be available in a formal or informal way and may also be limited or be very detailed and complete. Frequently the user is only able to provide a minimum amount of information. This happens when only a data set is supplied and nothing else is specified regarding the problem apart from defining the target variable. In this setting, it is the data analyst responsibility to determine which are the necessary and more suitable assumptions to be

made regarding the problem domain. This scenario may occur when a team of domains experts is simply not available or is too expensive. At the other extreme of information provided by the end-user, we can have complex tasks where a huge amount of information concerning the problem is provided. This information may include, for instance, the data set, specific preferences regarding the predictive performance of the models, costs associated with features, concrete rules that the models should obey, constraints on the features or target variable, noise levels of the features, among other. These tasks are hard to solve due to the difficulty in integrating all the existing domain information within the learning process. We use the notion of user information to represent the available raw domain knowledge.

Definition 3.2.1 (User Information) *The user information is all the knowledge that the end-user is able to provide regarding the predictive problem. This domain knowledge is represented by a n-tuple θ, with $n \geq 1$, and may be provided in an informal or formal way. The user information θ is always a non-empty tuple because it should contain at least the predictive task, $\mathcal{T}$, that includes the problem data $\mathcal{D}$ and the target variable Y, i.e.,*

$$\theta = \langle \mathcal{T}, \cdots \rangle \tag{3.1}$$

The user information may be a 1-tuple with information restricted to the available data and target variable ($\theta = \langle \mathcal{T} \rangle$), or may include, for instance, informal preferences regarding the predictive performance of some classes ($\theta = \langle \mathcal{T}$,"class *cancer* is more important than class *non-cancer*"$\rangle$). In some cases, the user may be able to completely specify a utility matrix UM_1. In this case, the provided user information is more formal: $\theta = \langle \mathcal{T}, UM_1(\hat{y}, y) \rangle$. The user information may also include more complex settings where more information is available, which may always be provided in a formal or informal way. Let us consider, for instance, that we are solving a predictive task involving the diagnosis of a certain disease. Let us also assume that a team of medical specialists is also collaborating. In this case, the user information can be much more complete, and may include the predictive task (data set and target variable), a utility matrix UM_2 provided by the field specialists, a set of rules to which the learned model must comply to, and also some information regarding the noise level of some features. This information could be specified as follows: $\theta = \langle \mathcal{T}, UM_2(\hat{y}, y)$,"when both features B and C are high, the target variable must be a negative value" "when features A and B are low the target variable must be high", "feature A has more than 50% of noise"$\rangle$.

It is clear that the existence of more domain knowledge has advantages. It allows the data analyst to better adjust the results of the learning process to the aspects of the predictive task that are more relevant to the end-user. On the other hand, when only the minimum domain knowledge is available, i.e., when θ is a 1-tuple and only the data set and target variable are provided, then the data analyst is forced to make assumptions for solving the predictive task. For instance, if the user does not state a differentiated interest across the target variable domain, then, the data analyst may assume that the end-user has a uniform

interest across this domain, meaning that all target variable values are equally important to the end-user.

The biggest obstacle when solving a predictive task with more user information, i.e., with $|\theta| > 1$ is related with the transformation of the provided information into a formal and usable format. Even when the amount of information provided is not large, the fact that it is often provided in a very informal way causes important problems to the data analyst. For instance, this can be easily observed when dealing with binary classification tasks where the most important class is under-represented. This information is usually provided in a very informal way and it is the data analyst responsibility to use this available information for determining the most suitable model for the task. The difficulties in using this informal user information are clearly mirrored on the difficulties associated with performance evaluation in these tasks. If the end-user preferences are not formally stated, then it is more difficult to determine how the models' performance should be evaluated.

The described difficulties show the need to convert the provided user information into a more formal setting. This formalisation is a responsibility of the data analyst that should interpret the user information transforming it into usable knowledge. We define this formalisation of the domain knowledge as the *learning context*.

Definition 3.2.2 (Learning Context) *The learning context of a predictive task is a k-tuple θ', with $k \geq 2$, that aggregates all the user information transforming it into formal and usable knowledge. θ' is a tuple that contains at least the two following elements: i) the predictive task $\mathcal{T}$ that includes the available data set and the problem target variable; and ii) the user preferences regarding the model predictive performance, i.e.,*

$$\theta' = \langle \mathcal{T}, \Pi, \cdots \rangle \tag{3.2}$$

where Π represents the formal definition of the predictive performance preferences.

The notion of learning context is used to reflect the transformation of the user information into a formal and operational setting. It requires the formal definition of at least two elements in θ': the predictive task, and the user preferences in terms of predictive performance. The first element of θ' is the most simple to obtain. The data analyst only has to consider eventual data pre-processing steps that may be necessary before learning can take place. The second element requires the formalisation of the user preferences concerning the predictive performance which is a more complex task. We will briefly explain how this task can be solved and will provide more details in the following sections.

All predictive tasks involve the definition of the predictive performance preferences, even when this is not directly provided by the end-user. We will distinguish three main types of user-provided information, θ: i) no information regarding predictive performance preferences is provided (**missing**); ii) only partial or informal information is provided (**partial**); or iii)

all the preferences are fully specified (**complete**). When the user information is provided as $\theta = \langle \mathcal{T} \rangle$ there is no extra information beyond the predictive task. In this case we say that the information regarding the user preferences is missing. A partial level of information is present whenever the user is able to provide some information but is not capable of fully specify his preferences. For instance, we have a partial level of information available when the end-user states that a certain class is more important than another. The following is an example of this situation: $\theta = \langle \mathcal{T}$, "class cancer is more important than class healthy"$\rangle$. We have a complete level of information when the user is able to fully specify these preferences which happens when, for instance, a utility matrix is provided, i.e., $\theta = \langle \mathcal{T}, UM_1(\hat{y}, y) \rangle$. Depending on the available level of information, the data analyst can use different tools to formally define the predictive performance preferences. We will now describe how this formalisation can be achieved.

We proposed to use two main concepts to formalise the end-user preferences: utility surfaces for regression tasks and utility matrices for classification tasks. These concepts were previously defined in Sections 2.3 and 2.4 (pages 18 and 21, respectively).

The utility matrix or surface concept allows to fully specify the end-user preferences regarding the possible predictions for each true target variable value. This setting corresponds to having available a **complete** level of information. In this case, the user information tuple already includes the formal definition of the user preferences required for the learning context, i.e., $\theta = \theta' = \langle \mathcal{T}, U_{inf} \rangle$, where U_{inf} represents the utility matrix or utility surface information. This means that, in this case, the issue of formalising the user preferences is already solved, and therefore the data analyst task is easier as he only has to deal with the data preparation issues.

The concept of relevance function is used when the end-user is only able to provide **partial** information and is applicable to both classification and regression problems.

Definition 3.2.3 (Relevance Function) *A relevance function, which we will denote by $\phi()$, is a function that maps the target variable into a scale of relevance in [0,1]:*

$$\phi : \mathcal{Y} \rightarrow [0, 1] \tag{3.3}$$

where 0 represents minimum relevance and 1 represents maximum relevance.

The relevance function can be directly provided by the end-user, or may be estimated by the data analyst. In the following sections we will discuss different strategies for obtaining the relevance function for a given predictive task. Figure 3.1 shows two examples of different relevance functions for the multiclass problem described by data set *glass*[1]. We can observe that different relevance scores are assigned to the different problem classes. In the left hand

[1]Data set available at `https://archive.ics.uci.edu/ml/datasets/glass+identification`

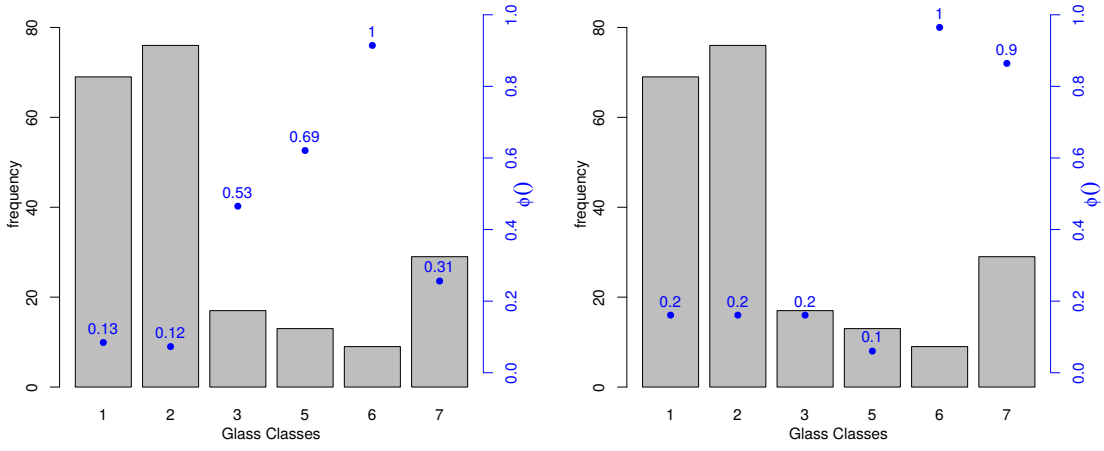


Figure 3.1: Examples of two possible relevance functions ϕ defined for the classification data set *glass* with the target variable frequency.

side graph, we observe that the least represented classes are the most important ones while on the right hand side the most relevant classes are class 6 and 7.

Figure 3.2 displays two examples of different relevance functions for the regression problem *autoPrice*[2]. The relevance function in the left side assigns a higher relevance to the regions with lower density of the target variable. The relevance function displayed on the right considers as the most relevant regions two particular ranges of the target variable (in the neighbourhood of the values of 0 and 20000) that are not associated with low density regions.

With this type of **partial** information, relevance functions can be used to formalise the user information into an operational learning context (θ'). For instance, the user information previously provided $\theta = \langle \mathcal{T}$, "class cancer is more important than class healthy"$\rangle$ can be transformed into the following learning context

$$\theta' = \langle \mathcal{T}, \phi(Y) = \begin{cases} 1 & \text{if } Y = cancer \\ 0.5 & \text{if } Y = healthy \end{cases} \rangle$$

Finally, we say that the level of information is **missing** when the user is not able to provide any information regarding the predictive performance preferences. In this case, there is no extra information from the end-user and therefore the data analyst is forced to make assumptions. The absence of information regarding the predictive performance preferences implies the assumption that the user preferences are uniform, i.e., all accurate predictions have the same benefit and all comparable errors have the same cost. This is the more reasonable assumption in the absence of any information provided by the end-user, and

[2]Data set available at `https://archive.ics.uci.edu/ml/datasets/automobile`

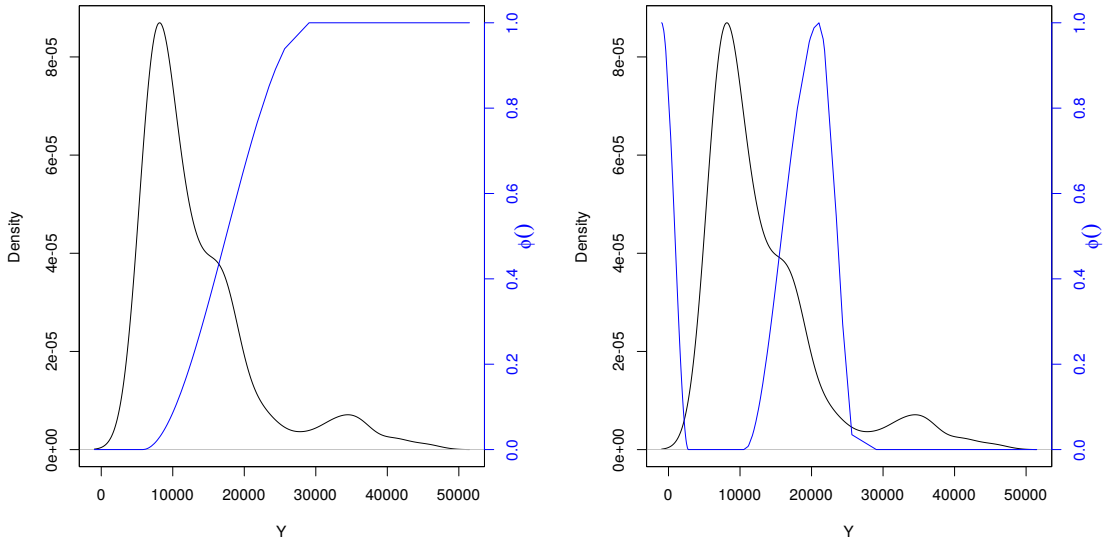


Figure 3.2: Examples of two possible relevance functions ϕ defined for the regression data set *autoPrice* with the target variable density.

leads to the following notion of uniform utility.

Definition 3.2.4 (Uniform Utility) *When the end-user has uniform preferences regarding the predictive performance of a model this means that all accurate predictions and all comparable errors have the same importance. In this case, we say that the predictive performance has a uniform utility, i.e., all accurate predictions have the same benefit and all comparable errors have the same cost. We represent this uniform utility as* $\mathcal{U}_{unif}$ *for both classification and regression tasks.*

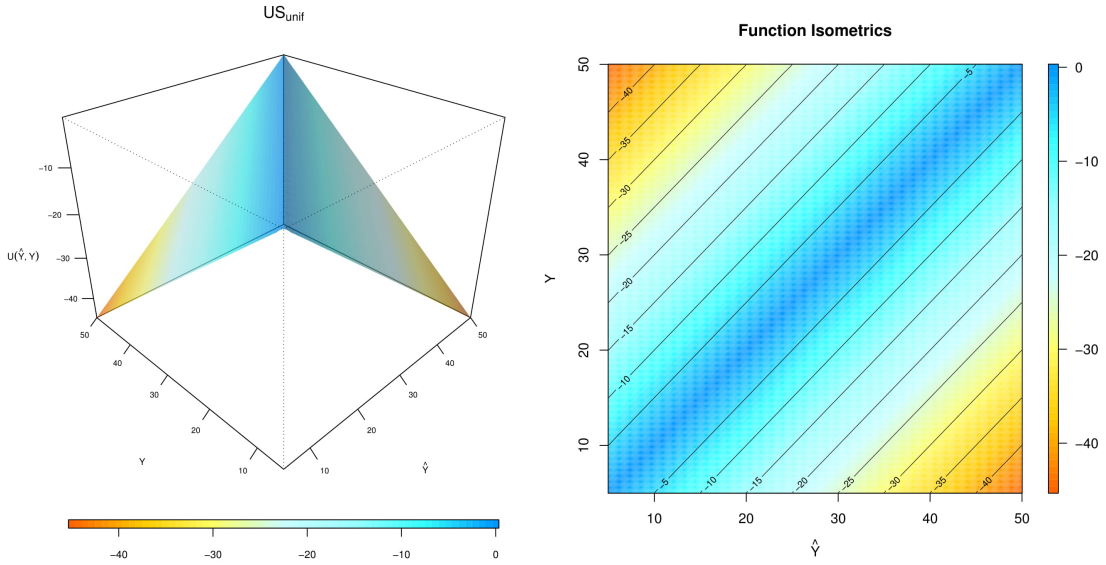
The uniform preferences of the end-user can be represented by a uniform utility matrix (cf. Definition 3.2.5), or by a uniform utility surface (cf. Definition 3.2.6), depending whether the problem being addressed is a classification or a regression task.

Definition 3.2.5 (Uniform Utility Matrix) *Let us consider a classification problem with uniform utility* $\mathcal{U}_{unif}$*. The utility matrix representing this setting is defined as follows:*

$$\mathcal{UM}_{unif}(\hat{y}, y) := \begin{cases} 0 & \text{if } \hat{y} = y \\ -1 & \text{if } \hat{y} \neq y \end{cases} \tag{3.4}$$

Definition 3.2.6 (Uniform Utility Surface) *Let us consider a regression problem with uniform utility* $\mathcal{U}_{unif}$*. The utility surface representing this setting is defined as follows:*

$$\mathcal{US}_{unif}(\hat{y}, y) := -|\hat{y} - y| \tag{3.5}$$



(a) Uniform utility surface $\mathcal{US}_{unif}$.

(b) Isometrics of the Uniform Utility Surface.

Figure 3.3: Illustration of a uniform utility surface $\mathcal{US}_{unif}$ and the corresponding isometrics.

To illustrate the previous concepts, Table 3.1 shows an example of a uniform utility matrix, $\mathcal{UM}_{unif}$, for a binary classification problem. In Figure 3.3a we show a uniform utility surface $\mathcal{US}_{unif}$ for a regression task, with Figure 3.3b showing the corresponding utility isometrics. By observing the utility isometrics of $\mathcal{US}_{unif}$, i.e., the lines that share the same utility value, we confirm that all errors with the same magnitude have the same costs and all accurate predictions have the same benefit.

Table 3.1: Uniform utility matrix $\mathcal{UM}_{unif}$ for a binary classification problem.

		True	
		A	B
Predicted	A	0	-1
	B	-1	0

Table 3.2 shows examples of the described levels of information of θ (the user information tuple) and corresponding learning context tuple, θ', for a classification problem.

The following example shows how the user information θ can be converted into formal knowledge by defining the learning context θ'. This example also illustrates the impact of using different user information on the learning task.

Example 3.2.7 (Learning with different θ: Breast Cancer Prediction) *Let us con-*

Information Level	θ (user information)	θ' (learning context)
Missing	$\langle \mathcal{T} \rangle$	$\langle \mathcal{T}, \mathcal{UM}_{unif}(\hat{y}, y) \rangle$
Partial	$\langle \mathcal{T}$, "class cancer is more important than class healthy" $\rangle$	$\langle \mathcal{T}, \phi(Y) = \begin{cases} 1 & Y = cancer \\ 0.5 & Y = healthy \end{cases} \rangle$
Complete	$\langle \mathcal{T}, UM_1(\hat{y}, y) \rangle$	$\langle \mathcal{T}, UM_1(\hat{y}, y) \rangle$

Table 3.2: Levels of information available concerning the user predictive performance preferences ($\mathcal{T}$ is the predictive task that includes the available data set and the problem target variable, $\phi(Y)$ represents the relevance function and UM_1 is a utility matrix).

sider the Breast Cancer Wisconsin (Diagnostic) data set[3]*. We will represent this data set by* $\mathcal{D}$*. The Breast Cancer Wisconsin data set was built from digitised images of a fine needle aspirate of patient's breast mass. The predictive task to solve is a binary classification problem where the goal is to obtain the patient's diagnosis into malignant or benign. The target variable,* ***diagnosis****, is nominal and may assume two different labels: M (malignant) or B (benign).* $\mathcal{D}$ *is composed by 30 numeric features extracted from the patient's images and are based on the characteristics of the cell nuclei present in each image. Table 3.3 displays the 10 characteristics observed from the images for this data set. For each of these characteristics, the mean, standard error and "worst" or largest of these features were computed, providing a total of 30 predictor variables that are used in this data set. In summary, the predictive task information* $\mathcal{T}$ *is composed by data set* $\mathcal{D}$ *with target variable* ***diagnosis****.*

The data set includes 569 examples and has 31 features (the 30 described above and an ID number). The target variable distribution is slightly imbalanced with 357 benign (B) and 212 malignant (M) cases.

Let us also consider the confusion matrices in Table 3.4 that represent the errors of three different models (m_1*,* m_2 *and* m_3*). Suppose that the data analyst goal is to select one of these models.*

We will observe the impact of solving this task while considering the following three different user information tuples that correspond to the three levels of information described before:

1. $\theta_1 = \langle \mathcal{T} \rangle$ *(****missing*** *information);*
2. $\theta_2 = \langle \mathcal{T},$ *"class M has twice the importance of class B"* $\rangle$ *(****partial*** *information);*
3. $\theta_3 = \langle \mathcal{T}, UM_3(\hat{y}, y) \rangle$*, where* $UM_3(\hat{y}, y)$ *is defined in Table 3.5 (****complete*** *information).*

[3]This data set is available through UCI [Dheeru and Karra Taniskidou, 2017] Machine Learning Repository at `http://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28diagnostic%29`.

Table 3.3: Description of Breast Cancer Data observed characteristics from the collected images.

Name	**Description**
radius	mean of distances from center to points on the perimeter
texture	standard deviation of gray-scale values
perimeter	cell nucleus perimeter
area	cell nucleus area
smoothness	local variation in radius lengths
compactness	$perimeter^2/area - 1$
concavity	severity of concave portions of the contour
concave points	number of concave portions of the contour
symmetry	symmetry of the cell nucleus
fractal dimension	"coastline approximation" - 1

Table 3.4: Confusion matrices of three models m_1, m_2 and m_3 built for the Breast Cancer problem (M: class malignant, B: class benign).

Model m_1		**True**	
		M	B
Predicted	M	152	59
	B	60	298

Model m_2		**True**	
		M	B
Predicted	M	110	1
	B	102	356

Model m_3		**True**	
		M	B
Predicted	M	159	80
	B	53	277

Table 3.5: UM_3: Utility matrix for the Breast Cancer prediction problem.

		True	
		M	B
Predicted	M	2	−1
	B	−7	1

Table 3.6 shows the scores obtained for the three models when using the three alternative user information settings. We will now explain how is the user information converted into the formal learning context, and the motivation for using the evaluation measures displayed.

Table 3.6: Scores obtained using different user information for the three different models described in Table 3.4.

		Scores			**Decision Rank**		
		m1	*m2*	*m3*	*m1*	*m2*	*m3*
θ_1	$L_{0/1}$	119	**103**	133	2	1	3
	$\mathcal{UM}_{unif}$	-119	**-103**	-133	2	1	3
θ_2	F_1	**0.719**	0.681	0.705	1	3	2
	$G-Mean$	**0.774**	0.719	0.763	1	3	2
	$CWA_{0.67}$	0.756	0.677	**0.759**	2	3	1
θ_3	UM_3	123	-139	**144**	2	3	1

In the setting corresponding to θ_1, the user only provides the predictive task and nothing else is specified. In this case, the data analyst must assume uniform preferences, i.e., all cases should be equally important and all mistakes equally serious. This means that, in this user information setting, it is as serious to fail a malignant prediction as it is to fail a benign case. Therefore, the goal of the predictive task when the user information θ is a 1-tuple, is to minimise a selected standard loss function such as the 0-1 loss ($L_{0/1}$) as defined in Equation 2.1 on page 11. If the data analyst is required to select between models m_1, m_2 and m_3, then model m_2 should be selected because this is the model with minimum loss. In effect, model m_2 only misclassifies 103 cases while models m_1 and m_3 misclassify 119 and 133 respectively. The minimisation problem described is equivalent to the problem of maximising the uniform utility. The user information θ_1 is formalised as $\theta'_1 = \langle \mathcal{T}, \mathcal{UM}_{unif} \rangle$. Using this formalisation, the data analyst goal is the maximisation of the utility. Model m_2 has a utility of -103 and models m_1 and m_3 have a utility of -119 and -133, respectively. Therefore, model m_2 should be selected because this model achieves a higher utility. As we can observe the problem of minimising the $L_{0/1}$ is equivalent to the problem of maximising the $\mathcal{UM}_{unif}$.

The second setting includes only a partial information. In this case the user expresses informally that class M is more important and is also able to quantify that it is twice more important than class B. In this case, the data analyst can formalise the user information using a relevance function. There are different ways to accomplish this. For this example, we will assume that the data analyst sets the relevance function as follows:

$$\phi(Y) = \begin{cases} 1 & if \;\; Y = M \\ 0.5 & if \;\; Y = B \end{cases} \tag{3.6}$$

This means that θ_2 is converted into $\theta'_2 = \langle \mathcal{T}, \phi(Y) \rangle$, where $\phi(Y)$ is defined by Equation 3.6. Although this function provides a formalization of the user preferences it still does not tell the analyst how to evaluate the predictive performance of the models. Several alternatives exist to evaluate the performance in these cases. It is the data analyst responsibility to determine which are the more suitable evaluation metrics to apply. A common option in problems where the end-user is more interested in one class is the use of the F_1 or $G-Mean$ evaluation metrics. The first is only focused on the most important class for the user while the second takes both classes into account. The CWA metric can also be suitable for this setting as it allows to weight the sensitivity and the specificity of the models being evaluated. In the described user information setting θ'_2, the data analyst could consider $w = 0.67$ which provides the sensitivity measure with a weight of 0.67 and the specificity measure with a weight of 0.33. This is motivated by the fact that, sensitivity is a measure focused on the positive class (M) while specificity is focused on the negative class (B) and the user assigned to class M twice the relevance of class B. As we can observe in Table 3.6 the three measures displayed for this case (F_1, $G-Mean$ and CWA with $w = 0.67$) lead to different ranking outcomes. Using the two first measures model m_1 achieves the best performance while using the last measure would lead to the selection of model m_3. Still, we must highlight that m_2 is always the worst performing model in this situation, as opposed to the previous setting where no user knowledge was available.

Finally, in the last user information setting θ_3 the user is able to fully specify the domain preferences by providing a utility matrix $UM_3(\hat{y}, y)$ displayed in Table 3.5. In this situation, there is a match between the user information and the learning context, i.e., $\theta_3 = \theta'_3$ and the data analyst has the formalisation problem solved. The best model under this learning context is the one that achieves the maximum utility according to the provided utility matrix. Models m_1, m_2 and m_3 obtain a total utility of 123, -139 and 144, respectively. This means that, for the learning context θ'_3, the best model is m_3.

As a conclusion we observe that the user information available is decisive in choosing the more suitable performance evaluation measures and consequently in the selection of the best model.

The previous example showed how the conversion of user information into learning context can be accomplished. Moreover, it also showed the importance of the second component of θ': the formalisation of the end-user preferences regarding the predictive performance. The notions of θ and θ' allow the formalisation of standard and non-standard tasks.

Definition 3.2.8 (Standard Predictive Task) *A predictive task is standard when the user information θ is a 1-tuple, or equivalently, the learning context θ' is a 2-tuple, having as second component uniform preferences over the models predictive performance, i.e., uniform utility. This means that we face a standard learning task when $|\theta| = 1$ or $\theta' = \langle \mathcal{T}, \mathcal{U}_{unif} \rangle$.*

In fact, in the previous example we observed that the evaluation of the models is equivalent when using both the standard evaluation metric ($\mathcal{L}_{0/1}$) and the uniform utility matrix ($\mathcal{UM}_{unif}$). The only difference between the two methods is related with the goal which changes from minimisation when using the standard evaluation measure to maximisation when using the uniform utility information.

Equivalently, it is also possible to define non-standard predictive tasks as follows.

Definition 3.2.9 (Non-standard Predictive Task) *A predictive task is non-standard when the user information θ is a n-tuple with $n > 1$.*

Non-standard predictive tasks involve the existence of more information than standard tasks. This information may be related with the end-user predictive performance preferences, or may be connected with other domain information such as noisy features or rules specified by domain experts to which the models must comply to.

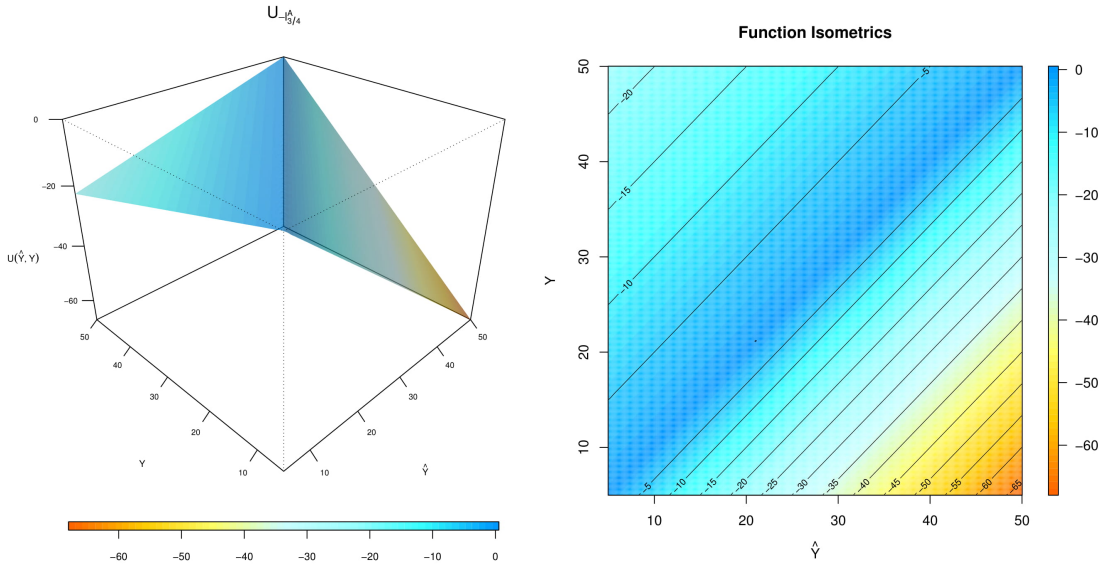
Predictive tasks that include information regarding non-uniform preferences on the predictive performance of the models are a special type of non-standard tasks named utility-based learning tasks (cf. Definition 3.2.10).

Definition 3.2.10 (Utility-based Learning Task) *A predictive task is a utility-based learning task when the learning context θ' is defined by a non-uniform relevance function or a non-uniform utility, i.e.,*

$$\theta' = \langle \mathcal{T}, \phi(Y) \rangle \vee \theta' = \langle \mathcal{T}, \mathcal{U} \rangle \tag{3.7}$$

where $\phi(Y)$ represents a non-uniform relevance function and $\mathcal{U}$ represents a non-uniform utility matrix or surface depending whether the problem is a classification or regression task, respectively.

This means that, whenever the user has information regarding the predictive performance preferences, this information can be transformed into formal and usable knowledge using



(a) Utility surface $\mathcal{U}_{-l^A_{3/4}}$.

(b) Isometrics of the Utility Surface $\mathcal{U}_{-l^A_{3/4}}$.

Figure 3.4: Utility surface $\mathcal{U}_{-l^A_{3/4}}$ and the corresponding isometrics.

either relevance functions or utility matrices/surfaces. Relevance functions provide less information while utility matrices/surfaces are a more complete picture of the user preferences. In both cases we face a utility-based learning task.

An example of utility-based learning tasks can be observed, for instance, when the user defines as his preferences non-standard loss functions, such as asymmetric loss functions. For instance, in a regression setting, it may be important to penalise more heavily the under predictions than the over-predictions of a model. Therefore, a possible scenario may involve the definition of the user information as follows: $\theta = \langle \mathcal{D}, l^A_{3/4} \rangle$, where l^A_α is defined by Equation 3.8 and represents the asymmetric absolute error function [Hernández-Orallo, 2013] also known as LIN-LIN function. Figure 3.4 shows the Utility Surface $\mathcal{U}_{-l^A_{3/4}}$ and the corresponding isometrics.

$$l^A_\alpha(\hat{y}, y) = \begin{cases} 2\alpha(y - \hat{y}) & \text{if } \hat{y} < y \\ 2(1-\alpha)(\hat{y} - y) & \text{otherwise} \end{cases} \tag{3.8}$$

Let us now consider some special cases of utility-based learning tasks. The definition of utility-based learning tasks that we have presented allows us to see *standard learning tasks as a special case of utility-based learning tasks*. In effect, a standard predictive task is just a utility-based learning task with a uniform utility $\mathcal{U}_{unif}$. In this context, we claim that all standard tasks are also utility-based learning tasks using a uniform utility matrix or a uniform utility surface.

Another special case of utility-based learning tasks are imbalanced domain learning tasks that occur when the defined relevance function is associated with some specific domain characteristics.

Definition 3.2.11 (Imbalanced Domain Learning Task) *A predictive task is an imbalanced domain learning task when the two following assertions are verified: i) the learning context is defined through a non-uniform relevance function; and ii) the relevance function is biased towards under-represented values of the target variable domain., i.e.,*

i) $\theta' = \langle \mathcal{T}, \phi(Y) \rangle$*, with* $\phi(Y)$ *non-uniform; and*

ii) $\phi(Y)$ *is higher on under-represented values of the target variable.*

This means that imbalanced domain predictive tasks are also a special case of utility-based learning tasks involving the definition of a non-uniform relevance function and also having an extra restriction concerning the distribution of the target variable.

Having defined a learning context, the data analyst still has to make other important decisions. Specifically, for a given learning context, many learning tools can be used and each may search for a model optimising different loss functions, that may or may not match the user preferences. Moreover, different performance estimation methodologies can be considered to select the best model for solving the predictive task. We will call these decisions the model selection context (cf. Definition 3.2.12). More precisely, the model selection context includes four different elements: the predictive task, that may be subject to some modification or not; the selected loss function (cf. Definition 2.3.1 provided in page 11); the modelling tools; and the estimation methodology.

Definition 3.2.12 (Model Selection Context) *The model selection context* ω *is a 4-tuple composed by: a (possibly modified) predictive task* $\mathcal{T}'$*; the loss function* $\mathbb{L}$ *that will guide the search process used during the learning of the models and that should be selected taking into account the user preference biases; the modelling tools to be considered, represented by* μ*; and the estimation methodology, denoted by* λ*. The modelling tools include information regarding the learning algorithm and respective set of parameters, while the estimation methodology includes the evaluation procedures and measures.*

$$\omega = \langle \mathcal{T}', \mathbb{L}, \mu, \lambda \rangle \tag{3.9}$$

It is the data analyst responsibility to define suitable $\mathbb{L}$, μ and λ for the predictive task under consideration. The model selection context is a crucial step because the model derived for solving the predictive task is obtained through a search process that is guided by the

optimisation of some loss function. To select this function the data analyst must take into consideration the learning context θ' of the task. In some applications it is not easy to select a loss function that bias the learning process to a model that is the best according to the user preferences that were formalised in the learning context. In these situations, a frequent option for the data analyst is to to modify the predictive task in such a way that achieving these goals becomes feasible. This is a strategy frequently used when dealing with imbalanced domain problems as described in Section 2.4.3.2.

Finally, the last step, also of the data analyst responsibility, consists of using the obtained model to produce predictions in a deployment context (cf. Definition 3.2.13).

Definition 3.2.13 (Deployment Context) *The deployment context δ is a n-tuple with $n \geq 2$, composed by all the information available at the moment of the deployment of the model built using the model selection context ω of the predictive task. The deployment context must contain, at least, the built model m and the deployment data Γ for which the data analyst is required to obtain predictions, i.e.,*

$$\delta = \langle m, \Gamma, \cdots \rangle \tag{3.10}$$

Data set Γ contains previously unseen observations for which predictions are required. More information may be provided in the deployment context. For instance, consider a task whose goal is to predict the volume of sales of a certain product. The data set provided in θ contains all the available information for a certain country. The data analyst uses this information to obtain the most suitable model. However, at deployment time, the user may receive a new data set Γ as well as the information that the model will only be applied in a particular region of the country. In this case, the model previously built may not be optimal. It is the data analyst responsibility to decide how to act in these cases. When $|\delta| > 2$ the data analyst must evaluate the newly received information and decide between one of the following options: i) apply the previously built model assuming that the new information provided will not have a relevant impact in the models' performance; ii) restart the process including in the user information θ the new information provided; or iii) adapt the built model to be suitable for the information that was provided. The first option may be selected when the new information is not important or will not impact the models' performance. This can also be the data analyst option when there is no time left to make readjustments and the new predictions are required without any delay. The second option can be followed when time is not an issue and the data analyst is focused on the models' performance rather than making fast predictions when the deployment context is known. In this case, the complete procedure can be rethought to incorporate the new information. The user information θ is completed with the more recent information and then the data analyst should rebuilt θ' and ω to derive the optimal model. Finally, the last option involves re-adapting the previously built model to reflect the newly provided information. This process of modifying a model after it is built,

adapting it to a new context is known as reframing (e.g. Hernández-Orallo [2014], Hernández-Orallo et al. [2016]). An example of this setting can be observed when, at deployment time, the data analyst is informed that over predictions are more serious than under-predictions. The previously built model m can be adapted to this new setting by applying a constant positive shift to the obtained predictions. This would inflate the models' predictions which would allow to obtain less over-predictions (which are more serious) at the cost of obtaining more, but less important, under-predictions.

Tables 3.7 and 3.8 show examples of the main groups of information that we have used to characterise a predictive task, namely, of user information (θ), the learning context (θ') and the model selection context (ω), for classification and regression tasks, respectively. We have not included the deployment context in these tables and assumed the simpler deployment context containing only a new data set Γ and the model.

The first classification examples in Table 3.7 describe a standard classification task. We can observe that the data analyst may select either the $\mathcal{L}_{0/1}$ or the uniform utility matrix $\mathcal{UM}_{unif}$ as the loss function to use. We also highlight that in this case the predictive performance can be characterised through either accuracy or utility-based metrics (total utility or NMU described in Section 3.3.2 in page 88). The second example in the table represents a situation of utility-based classification where the user provides the full information regarding the predictive performance preferences through a certain utility matrix UM_2. The third and fourth examples of Table 3.7 represent situation of imbalanced classification tasks. We observe that in these cases the user provides an informal and partial knowledge regarding the problem predictive performance preferences. For each one of these cases the data analyst derives a relevance function. We observe that the data analyst decides to optimise a standard loss function but in both cases the predictive task is changed. In particular in these cases the change applied is a data pre-processing method that allows to balance the fraud and non-fraud examples in the available data. This change enables the use of standard learning methods (random forests and neural networks) that optimise a standard loss function. However, in the evaluation of the performance of the obtained models measures that are suitable to the problem goals should be selected.

The regression tasks shown in Table 3.8 also include examples of standard, utility-based and imbalanced regression tasks. The first example in this table corresponds to a standard regression task. We observe that the data analyst assumes that the end-user has uniform preferences regarding the predictive performance of the models. In this case, the data analyst may choose between the minimisation of the $\mathcal{L}_{\mathcal{AE}}$, or the maximisation of $\mathcal{US}_{unif}$. In this situation the performance evaluation should be carried out through the MAE measure or through utility-based measures such as the total utility or the MU (cf. Equation 2.19 in page 24). The second example in Table 3.8 concerns a utility-based regression task where the end-user provides the full utility information through a utility surface US_2. This is a

simple task from the data analyst perspective because no effort is required to formalise the predictive performance preferences. In this case we observe that the data analyst selected a special purpose modelling tool, and evaluates the performance though utility-based measures. In the third example, the user information θ provided is informal. Apart from providing $\mathcal{T}_3$, which contains the available data and the problem target variable, the end-user is only able to specify that he is more interested in the high target variable values. The data analyst should transform this informal knowledge into a formal form. This is accomplished though the definition of a relevance function ϕ_3 that is displayed in Figure 3.5. We must highlight that the function ϕ_3 is only one possible definition. Given the informal knowledge provided the data analyst could have defined a different relevance function as long as a higher relevance is assigned to the high target variable values. Figure 3.5 also shows the boxplot of the target variable values. In this case, it is clear that we face an imbalanced regression problem because the most relevant values are poorly represented in the available data. In this example, the data analyst has pre-processed the available data set transforming the predictive task $\mathcal{T}_3$ into $\mathcal{T}_3'$. Regarding the performance evaluation the data analyst has selected the F_1^{ϕ} measure that is suitable for imbalanced regression problems. The fourth example in Table 3.8 displays a utility-based regression task with an informal level of information provided by the end-user. In this case it is only specified that under-predictions are worse than over-predictions. This requirement could be formalised by multiple utility surfaces. In this example, the data analyst selected to use a utility surface $\mathcal{U}_{-l^A_{3/4}}$ (displayed in Figure 3.4) based on an asymmetric loss function (cf. Equation 3.8) that penalises more under-predictions than over-predictions. For this example, the adequate metrics to evaluate the performance of the models are utility-based measures. The last example shows an imbalanced regression task for which the end-user is able to fully specify the relevance function ϕ_5. This represents an imbalanced regression task because ϕ_5 is not uniform and the target variable values with higher relevance correspond to low density regions in the available data set. The data analyst addressed this task by modifying the data set which allowed the use of a standard learning algorithm. For the performance evaluation the data analyst decided to use the F_1^{ϕ} measure.

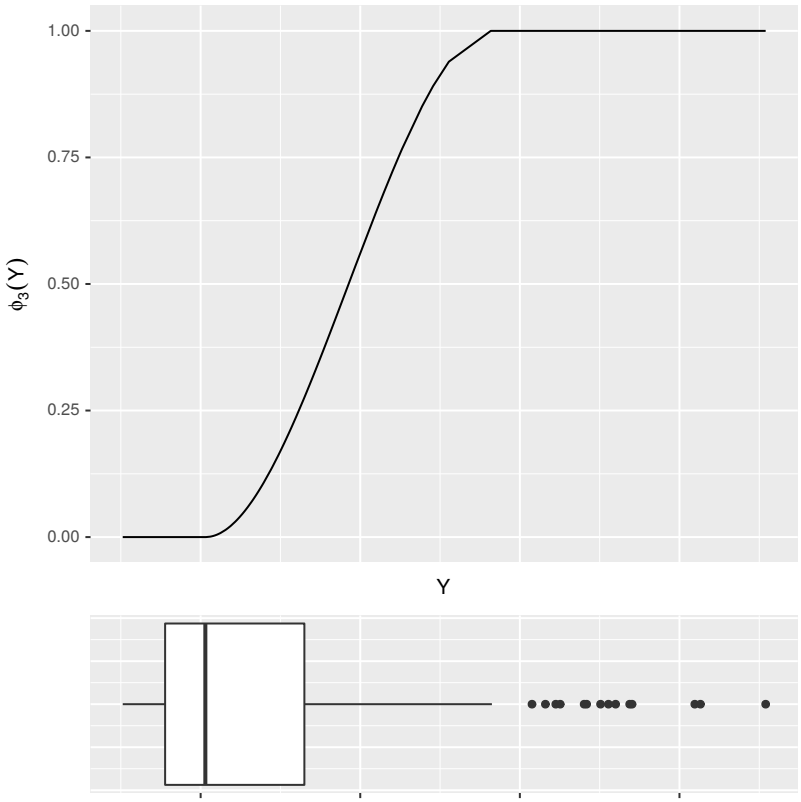


Figure 3.5: Relevance function ϕ_3 derived by the data analyst for the example with ID 3 in Table 3.8.

Table 3.7: Examples of classification tasks with the characteristics decomposed according to the defined predictive analytics framework.

ID	θ	θ'	ω				Task
			$\mathcal{T}'$	$\mathbb{L}$	μ	λ	
1	$\langle\mathcal{T}_1\rangle$	$\langle\mathcal{T}_1, \mathcal{UM}_{unif}\rangle$	$\mathcal{T}_1$	$\mathcal{L}_{0/1}(\hat{y}, y)$ or $\mathcal{UM}_{unif}$	decision tree; $minsplit = \{10, 20, 30\}$; $maxdepth = \{10, 15, 20, 25\}$	2 repetitions of 5 fold CV; Accuracy, Total Utility or NMU	Standard Classification
2	$\langle\mathcal{T}_2, UM_2\rangle$	$\langle\mathcal{T}_2, UM_2\rangle$	$\mathcal{T}_2$	$UM_2(\hat{y}, y)$	MetaCost algorithm with decision tree as base learner; default parameters	10-fold stratified CV; Total Utility, Mean Utility or NMU	Utility-based Classification
3	$\langle\mathcal{T}_3$, class "fraud" is the most important$\rangle$	$\langle\mathcal{T}_3$, $\phi(\text{“}fraud\text{”}) = 1$; $\phi(\text{“}non-fraud\text{”}) = 0.2\rangle$ Note: "fraud" is under-represented in the data set provided in $\mathcal{T}_3$	$\mathcal{T}'_3$	$\mathcal{L}_{0/1}(\hat{y}, y)$	random forest; $mtry = \{5, 7, 9\}$; $ntree = \{1000, 1500, 2000\}$	100 repetitions of .632 bootstrap procedure; F_1	Imbalanced Classification
4	$\langle\mathcal{T}_4$, "fraud" is two times more important than "non-fraud"$\rangle$	$\langle\mathcal{T}_4$, $\phi(\text{“}fraud\text{”}) = 1$; $\phi(\text{“}non-fraud\text{”}) = 0.5\rangle$	$\mathcal{T}'_4$	$\mathcal{L}_{0/1}(\hat{y}, y)$	neural network; $size = \{2, 4, 6, 8, 10\}$; $decay = \{0, 0.01, 0.05, 0.1\}$	5-fold stratified CV; $G-Mean$ or CWA with $w = 0.67$	Imbalanced Classification

Table 3.8: Examples of regression tasks with the characteristics decomposed according to the defined predictive analytics framework.

ID	θ	θ'	ω				Task
			$\mathcal{T}'$	$\mathbb{L}$	μ	λ	
1	$\langle \mathcal{T}_1 \rangle$	$\langle \mathcal{T}_1, \mathcal{US}_{unif} \rangle$	$\mathcal{T}_1$	$\mathcal{L}_{AE}(\hat{y}, y) = \lvert y - \hat{y} \rvert$ or $\mathcal{US}_{unif}(\hat{y}, y)$	random Forest; $mtry = \{6, 8, 10\}$; $ntree = \{500, 1000, 1500, 2000\}$	2 repetitions of 10 fold CV; MAE or Total Utility or MU	Standard Regression
2	$\langle \mathcal{T}_2, , US_2 \rangle$	$\langle \mathcal{T}_2, US_2 \rangle$	$\mathcal{T}_2$	$US_2(\hat{y}, y)$	special purpose modelling tool that optimises US_2	5 fold CV; Total Utility or MU	Utility-based Regression
3	$\langle \mathcal{T}_3$, "high target variable values are the most important" $\rangle$	$\langle \mathcal{T}_3, \phi_3 \rangle$Note: "higher target variable values are scarce"	$\mathcal{T}_3'$	$\mathcal{L}_{SE}(\hat{y}, y) = (\hat{y} - y)^2$	standard SVM with RBF kernel; $\gamma = \{0.1, 1, 10\}$; $C = \{10, 20, 30\}$	200 repetitions of .632 bootstrap procedure; F_1^{ϕ}	Imbalanced Regression
4	$\langle \mathcal{T}_4$, "under-predictions are worse than over-predictions" $\rangle$	$\langle \mathcal{T}_4, \mathcal{U}_{-l^A_{3/4}} \rangle$	$\mathcal{T}_4$	$l^A_{3/4}(\hat{y}, y) = \begin{cases} \frac{3}{2}(y - \hat{y}) & , \hat{y} < y \\ \frac{1}{2}(\hat{y} - y) & , \hat{y} \geq y \end{cases}$	special purpose learning algorithm optimising $l^A_{3/4}$	10 repetitions of 10-fold CV; total utility or MU	Utility-based Regression
5	$\langle \mathcal{T}_5, \phi_5 \rangle$	$\langle \mathcal{T}_5, \phi_5 \rangle$Note: values with high ϕ_5 correspond to low density regions in the data set contained in $\mathcal{T}_5$	$\mathcal{T}_5'$	$\mathcal{L}_{AE}(\hat{y}, y) = \lvert \hat{y} - y \rvert$	linear regression model	2 repetitions of 5-folds CV; F_1^{ϕ}	Imbalanced Regression

3.3 Utility-based Regression Challenges

The problem of utility-based learning was formally presented in the previous section (cf. Definition 3.2.10). In this section we discuss the open challenges of a sub-class of these problems: utility-based regression tasks. This is a less explored problem and still has several important open issues to be solved. Ideally, in utility-based regression tasks, the user should provide a utility surface expressing the domain knowledge, i.e., the user information θ should be a 2-tuple containing the available data set and respective target variable ($\mathcal{T}$), as well as the end-user non-uniform predictive performance preferences represented by a non-uniform utility surface ($\mathcal{US} \neq \mathcal{US}_{unif}$). As we have mentioned before, in some cases the user may only be able to provides a relevance function or a even more informal information regarding the predictive performance preferences. This is frequently the case in imbalanced domains.

Having formalised the utility information in θ' we can use it to calculate a utility score when evaluating the models. The models with higher utility are preferred over the ones with a lower utility score. In this context, the utility information is a key component of these problems where the goal is to maximise the utility.

Researchers dealing with utility-based regression tasks must face the following three main challenges: i) obtaining the utility information; ii) deciding how to assess the performance on the models; and iii) how to obtain models that optimise utility.

The first challenge involves the formalisation of user preferences as a utility surface. This is particular challenging for regression tasks where we have potentially an infinite number of target variable values. Ideally, the end-user should be helped in the process of deriving a utility surface that matches his goals. When a utility surface ($\mathcal{US}$) is fully specified in θ' the task of selecting a suitable loss function $\mathbb{L}$ for the model selection context ω becomes easier. The second challenge regards the need to consider a suitable evaluation framework, i.e. the definition of the λ component of the model selection context ω. Finally, the third challenge has to do with the current lack of solutions to obtain models that are able to optimise utility. This is related with the selection of suitable modelling tools μ in the model selection context ω. Although there are some solutions for this problem they are focused on classification tasks while for regression tasks they are still poorly explored. We can observe that the described challenges cover different important groups of information that characterise the learning task, namely, they impact the formalisation of θ', $\mathbb{L}$, μ and λ.

In this section we will provide solutions for solving the two first challenges. The third challenge is discussed in Chapter 4 where we present new solutions for optimising the utility on utility-based regression tasks.

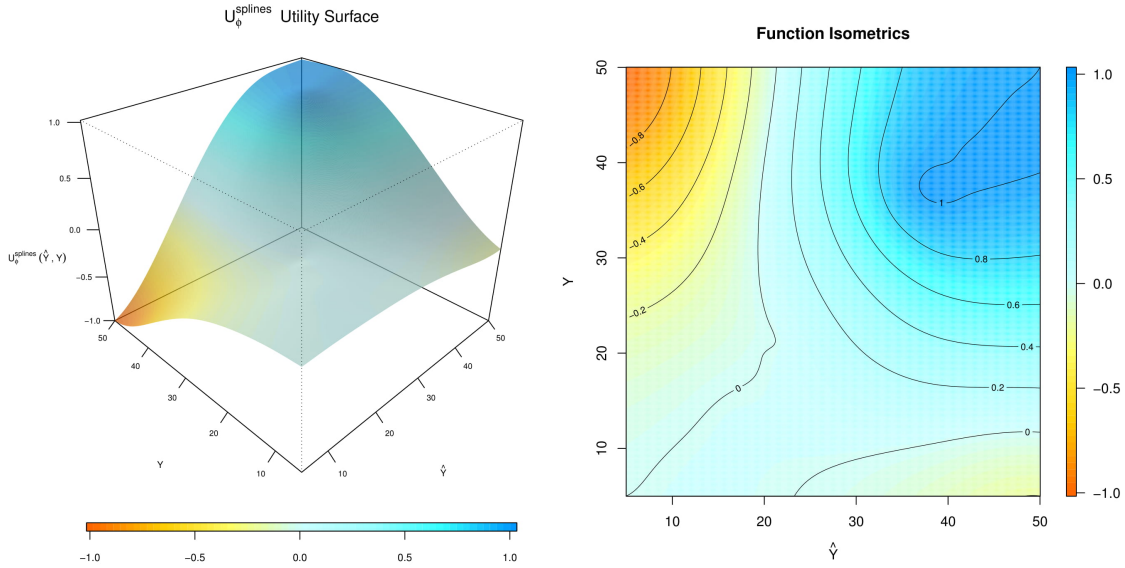
3.3.1 The Challenge of Obtaining the Utility Surface

As we have mentioned, the lack of utility information in classification problems is an important obstacle to the development of solutions for these tasks. Teams of domain experts have frequently high associated costs. Moreover, it is difficult to find available domain experts. This poses an important challenge when trying to define the utility matrix for a given practical application.

Although already being challenging in the context of classification tasks, this is even harder when we consider utility-based regression tasks. This happens because the user is required to specify the utility assigned to all possible combination of pairs $(\hat{y}, y)$ of predicted and true target variable values. The continuous nature of the target variable in regression problems, makes this task extremely complex. Still, only when a mapping between these pairs and a utility score is available it is possible to effectively train and assess utility-based models. In this section we will focus on solutions for deriving this utility information for regression tasks. This will help to provide a more formal user information θ which in turn also alleviates the data analyst task when building the learning context θ'. Moreover, this utility surface information can also be used by the data analyst as the loss function ($\mathbb{L}$) to use in the models' optimisation procedure.

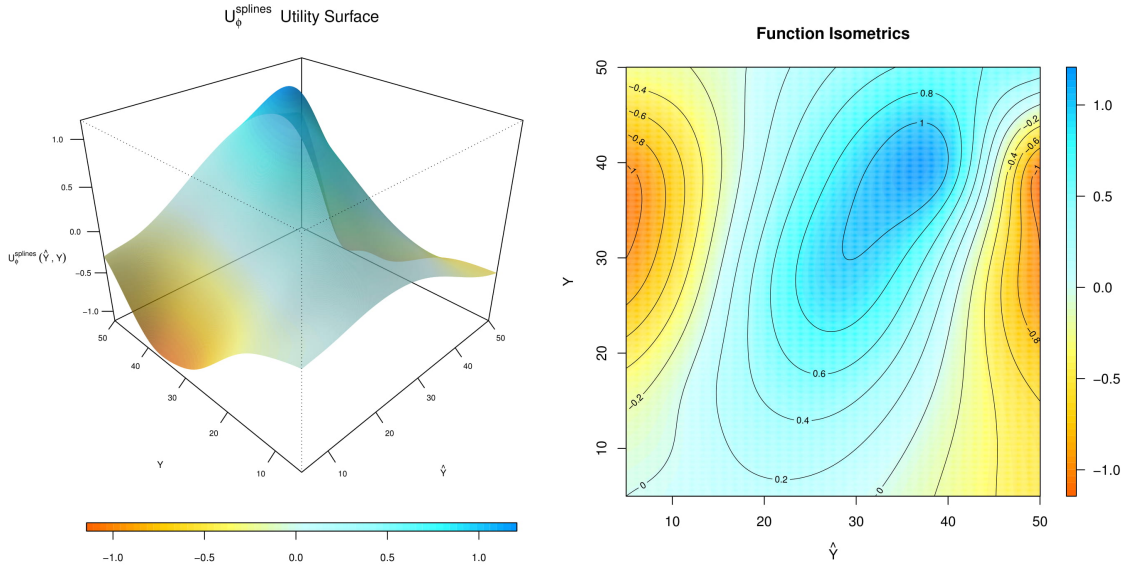
Typically, the utility surface definition is problem dependent. Moreover, different utility surfaces may be preferable depending on the users' goals. To illustrate the impact that the users' goals may have in the utility surface let us consider a data set where the target variable expresses the degree of a certain illness. From a human health perspective, the higher the illness degree the more important is its diagnosis. The higher values of the target variable correspond to the most severe and complicated situations and therefore are the most important ones. Figure 3.6a shows a utility surface that could be suitable for this problem considering the described point of view. Figure 3.6b shows the isometrics corresponding to the utility surface in Figure 3.6a.

However, this same data set, from a hospital manager perspective, could have a different utility surface. In fact, when expenses with treatments and exams are taken into account, the most important cases may be different. Mistakes in certain values may imply unnecessary and expensive exams, while for other values this may be less important. Let us assume that, for the described illness, having a good predictive performance for the values in the range $[30, 40]$ is the most important. This means that an accurate prediction for values in $[30, 40]$ leads to the high benefits while a poor performance in these values leads to expensive treatments that were not needed. This means that the hospital incurs into high costs if these cases are incorrectly predicted. Figure 3.7 shows a utility surface and the corresponding isometrics that could be preferable for this other situation. As we can observe, the utility surfaces of both perspectives are very different. A model considered good for predicting under the first setting could display a poor performance for the second setting and vice-versa.



(a) Utility surface. (b) Utility surface isometrics.

Figure 3.6: Illustration of a utility surface defined for a data set expressing a illness severity that could be suitable from a medical perspective.



(a) Utility surface. (b) Utility surface isometrics.

Figure 3.7: Illustration of a utility surface defined for a data set expressing a illness severity that could be suitable from a hospital manager perspective.

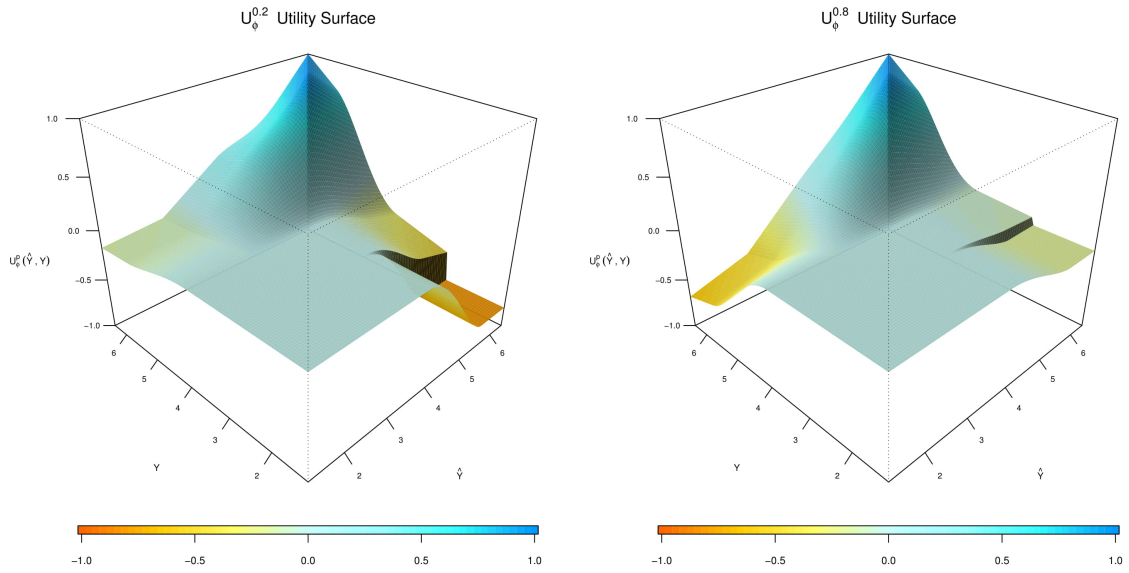
The responsibility of defining an adequate utility surface for the problem lies with the end-user. Still, given that this is a hard task, Ribeiro [2011] proposed a solution that decreases the effort required from the user. The solution of Ribeiro [2011] allows to automatically obtain both a relevance function and a utility surface for regression tasks making assumptions that are reasonable for a particular subclass of utility-based learning problems known as imbalanced domain problems. Moreover, this method also assumes that the most important values of the target variable can only be located on the extremes of the target variable distribution (i.e. either high or low values). Ribeiro [2011] proposed an automatic method that establishes relevance scores assuming that the rarest extreme values are likely to be the most important ones. Alternatively, the relevance function can also be obtained by the interpolation of relevance values given at specific points. The automatic method for obtaining the relevance function assumes that the relevance is inversely proportional to the target variable probability density function. The proposed automatic method provides an estimate of the relevance function $\phi(Y)$ by assigning more relevance to the rare and most extreme values. This is achieved by using the estimated quartiles and the inter-quartile range of the target variable distribution.

Ribeiro [2011] also proposed a method for obtaining the utility score of a problem using the notions of costs and benefits (cf. Equation 3.11). A utility score for each pair of values $(\hat{y}, y)$ is calculated by taking into account both the error measured through a given loss function and the relevance of y and $\hat{y}$. More precisely, this method the utility of the predictions of a regression model is given by a net balance between the benefits and costs of the predictions,

$$\begin{aligned} U^p_\phi(\hat{y}, y) &= B_\phi(\hat{y}, y) \; - \; C^p_\phi(\hat{y}, y) \\ &= \phi(y) \cdot (1 - \Gamma_B(\hat{y}, y)) \; - \; \phi^p(\hat{y}, y) \cdot \Gamma_C(\hat{y}, y) \end{aligned} \tag{3.11}$$

where functions $B_\phi(\hat{y}, y)$ and $C^p_\phi(\hat{y}, y)$ represent the benefits and the costs incurred when predicting $\hat{y}$ for a true value y, respectively. The benefit of a prediction ($B_\phi(\hat{y}, y)$) is a proportion of the relevance of the true value. If we have a perfect prediction (i.e. equal to the true value), then the benefit is maximum and equal to this relevance. This is captured by the expression $\phi(y) \cdot (1 - \Gamma_B(\hat{y}, y))$, where $\Gamma_B(\hat{y}, y)$ is a bounded loss function that normalises the standard loss function ($L(\hat{y}, y)$) into a $[0, 1]$ scale, so that after $L(\hat{y}, y)$ reaches a certain value the bounded loss is maximum (1). Regarding the cost of a prediction ($C^p_\phi(\hat{y}, y)$) this is calculated as a proportion of the maximum cost that is defined as a weighted average between the relevance of the true value and the relevance of the predicted value. The cost is obtained through expression $\phi^p(\hat{y}, y) \cdot \Gamma_C(\hat{y}, y)$. Function $\phi^p(\hat{y}, y)$ provides the weighted relevance average and uses parameter p to defined the weights between the two relevance values (0.5 gives equal importance to both). Function $\Gamma_C(\hat{y}, y)$ is a bounded loss function in the scale $[0, 1]$, similarly to $\Gamma_B(\hat{y}, y)$.

In the proposed framework the user is able to obtain different utility surfaces by changing



(a) Utility surface with $p = 0.2$.

(b) Utility surface with $p = 0.8$.

Figure 3.8: Utility surfaces obtained automatically with the method proposed by Ribeiro [2011].

the parameter p. This parameter has an impact on the types of errors that are more or less penalised in the utility surface. If parameter p is set to 0.5, then the same weight is assigned to all types of errors. This is similar to the process of assigning more costs to false positives, false negatives or both in classification tasks. Setting p to a value above 0.5 means that missing an important prediction has a high cost. On the other hand, if the user defines a value of p below 0.5, then to predict as important a value that is not important has the higher costs. Figure 3.8 displays two utility surfaces obtained for different values of p. In Figure 3.8a parameter p was set to 0.2, while Figure 3.8b shows a utility surface with p set to 0.8.

Although being automatic, the proposed method is only applicable for obtaining the utility surface of some particular utility-based problems: imbalanced domains problems where the important cases are located at the extremes of the target variable distribution. To overcome this limitation we propose a different method for obtaining utility surfaces. This method requires some user input but is also more general in the sense that it allows to derive any utility surface.

Essentially, the proposed method applies a selected point interpolation technique on a set of user provided points. This way a full utility surface is obtained and can be visualised. The end-user can always adapt the results by changing the selected interpolation method or adding/removing interpolating points when the surface does not match the initial expectations. The end-user is required to specify: i) the utility of some pairs $(\hat{y}, y)$ for which the

utility score is known; and ii) the interpolation method to use. The complete information of a utility surface is generated by applying the interpolation technique selected to the utility information provided. This method advantage lies on the flexibility offered to the user that is able to obtain specifically tailored utility surfaces with a minimal amount of supplied information.

We have implemented this method in the `UBL` R package [Branco et al., 2016a] where the user may select among one of four different interpolation methods: bilinear [W. S. Cleveland and Shyu, 1991], splines [Lee et al., 1997], idw [Cressie, 1993] and kriging [Cressie, 1993]. Table 3.9 briefly describes these interpolation methods and corresponding used R packages.

Method	**Description**	**R package**
bilinear	local fitting of a polynomial surface of degree 1	`stats` [R Core Team, 2018]
splines	multilevel B-splines	`MBA` [Finley and Banerjee, 2014]
idw	inverse distance weighted interpolation	`gstat` [Pebesma, 2004]
kriging	automatic kriging	`automap` [Hiemstra et al., 2008]

Table 3.9: Interpolation methods, and corresponding R packages, for obtaining different utility surfaces.

The goal of our proposal is to allow a higher flexibility for the utility surface definition while maintaining the simplicity for the user. Let us see some examples of utility surfaces obtained using the proposed interpolation method.

Example 3.3.1 (Obtaining utility surfaces based on the interpolation method) *Let us consider the regression data set named LNO2Emissions*[4]*. The target variable (LNO2) represents hourly measured values of the logarithm of the concentration of NO2 (particles) in Oslo, Norway, between October 2001 and August 2003. The data set has seven features that include information on the traffic, temperature, wind, hour and day. We will provide three cases for illustrating the use of the interpolation method proposed.*

Case 1 *Let us suppose that the end-user is able to provide, for this predictive task, the utility information described in Table 3.10. Let us also assume that he is not certain regarding the interpolation method that should be applied.*

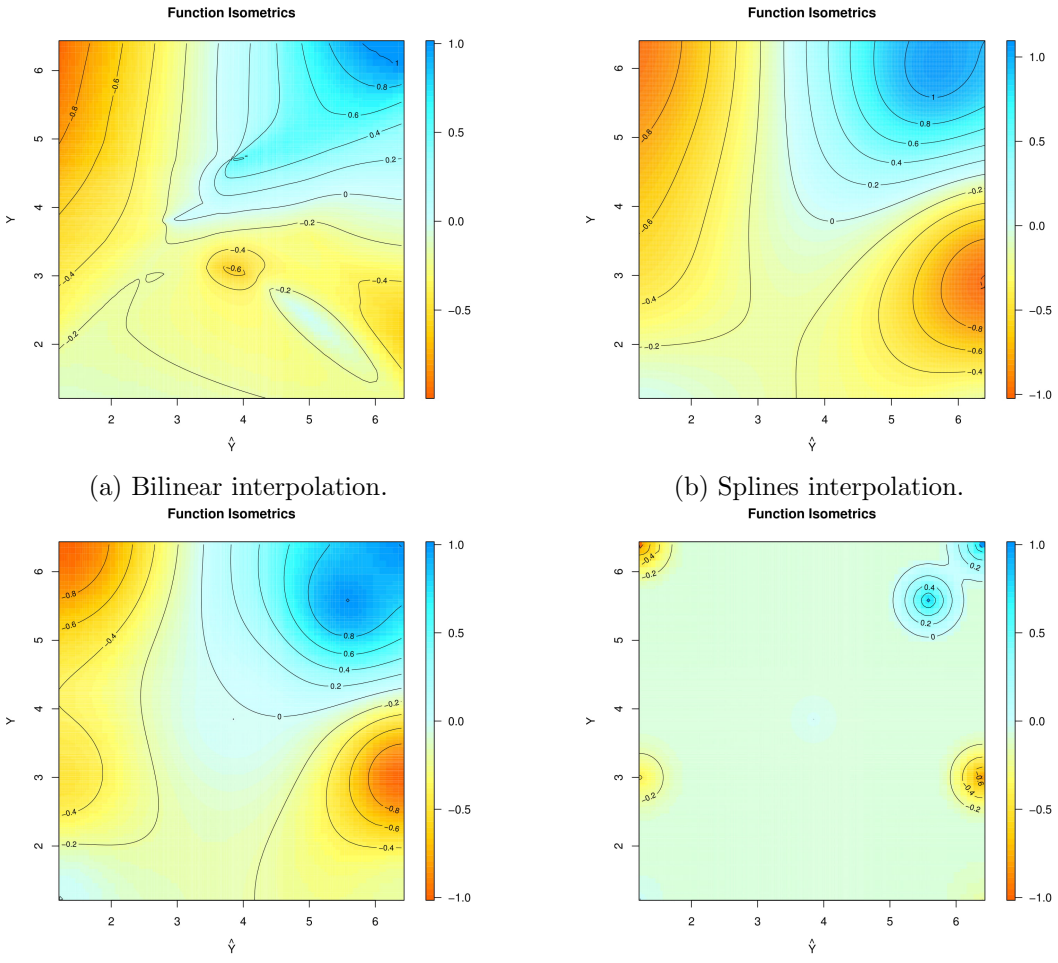
By observing the utility surfaces obtained using different interpolation methods, the end-user may verify which method is more suitable for the utility surface that he wants to generate. Figures 3.9a to 3.9d show the isometrics of the utility surfaces generated using the information in Table 3.10. These figures show that the same set of points generates very different surfaces. We must highlight that the "splines" interpolation method allows to

[4] A set of 500 examples from a study relating air pollution with traffic volume and meteorological variables. The data is available from the StatLib Datasets Archive: `http://lib.stat.cmu.edu/datasets/`.

Table 3.10: Utility surface information provided by the user for Case 1.

y	$\hat{y}$	$U(\hat{y}, y)$
1.22378	1.22378	0
3.84802	3.84802	0
5.58237	5.58237	1
1.22378	6.39509	-0.2
6.39509	1.22378	-1
3	1.22378	-0.5
3	6.39509	-1

obtain a smoother surface, while the "kriging" requires the specification of a higher number of points, and therefore is more suitable for situations where there is a greater amount of available utility information.



(a) Bilinear interpolation.

(b) Splines interpolation.

(c) Idw interpolation.

(d) Kriging interpolation.

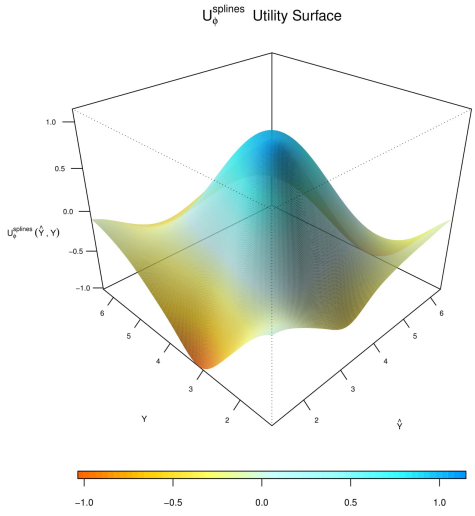
Figure 3.9: Isometrics obtained using different interpolation methods using the information provided by the user described in Table 3.10.

Case 2 *In this case we will consider that the user specifies the utility of a set of 10 points, assigning to the values of 2.5 and 3.5 the maximum utility score of 1, while 0 was assigned to the extremes of the target variable. The utility information provided by the user is displayed in Table 3.11.*

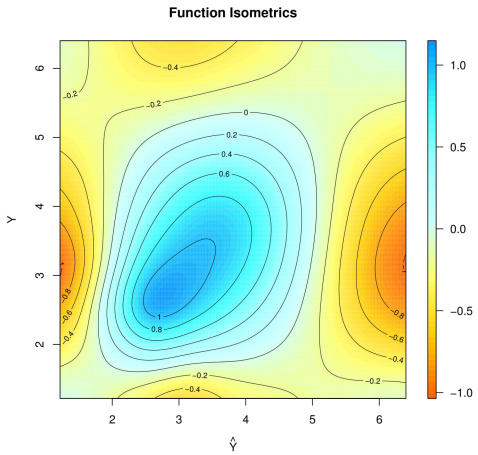
Table 3.11: Utility surface information provided by the user for Case 2.

y	$\hat{y}$	$U(\hat{y}, y)$
2.5	2.5	1
3.5	3.5	1
1.22378	1.22378	0
6.39509	6.39509	0
1.22378	6.39509	-0.1
6.39509	1.22378	-0.1
3	6.39509	-1
3	1.22378	-1
1.22378	3	-0.5
6.39509	3	-0.5

Figure 3.10 shows the utility surface and corresponding isometrics generated using the described information and the "splines" interpolation method. We selected this method due to its ability of generating a smoother surface, but any other method could be used. We highlight that, although some effort was required from the user, with a minimum of information it is easy to derive a domain specific utility surface.



(a) Utility surface.



(b) Isometrics.

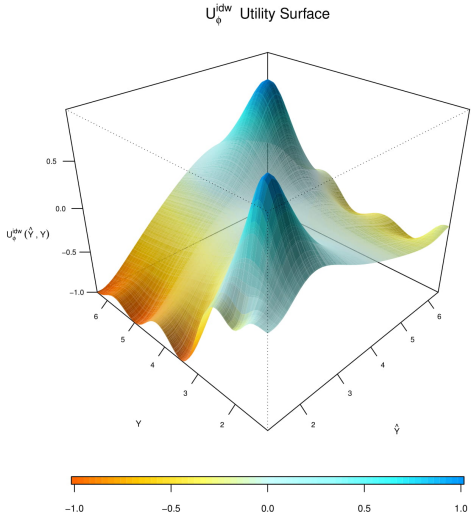
Figure 3.10: Utility Surface and isometrics generated with splines interpolation method using the information provided in Table 3.11 for Case 2.

Case 3 *In this case we show a surface built with more domain knowledge available and a different interpolation method. In this case the user provides 13 points for interpolation. Two non-extreme target variable values (2 and 5) are considered relevant. The information supplied for this case is shown in Table 3.12.*

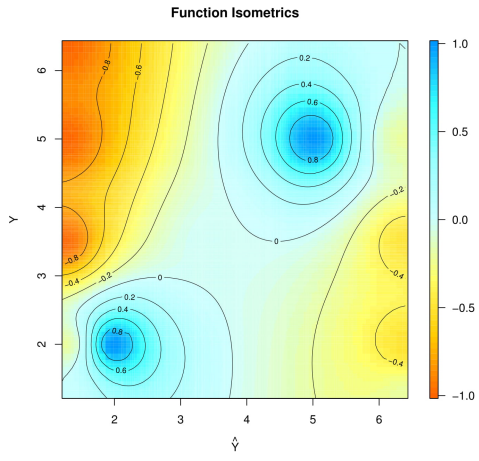
In this case the "idw" interpolation method was selected to derive the utility surface. Figure 3.11 displays the utility surface obtained and the corresponding isometrics.

Table 3.12: Utility surface information provided by the user for Case 3.

y	$\hat{y}$	$U(\hat{y}, y)$
2	2	1
5	5	1
3.5	3.5	0
1.22	1.22	0
6.39	6.39	0
1.22378	6.39509	−0.2
2	6.39509	−0.5
3.5	6.39509	−0.5
5	6.39509	−0.2
2	1.22378	−0.2
3.5	1.22378	−1
5	1.22378	−1
6.39509	1.22378	−1



(a) Utility surface.



(b) Isometrics.

Figure 3.11: Utility Surface and isometrics generated with idw interpolation method using Table 3.12 information for Case 3.

The proposed tool provides an easy way for solving the problem of deriving the utility surface

requiring a minimum of user effort. The user is able to obtain different utility surfaces simply by changing the provided points. It may happen that the utility surface derived does not correspond to the user preferences. In that case, it is easy to adjust the utility surface, being only necessary that the user corrects the introduced points or adds new points for adjusting the surface till it meets his requirements.

3.3.2 The Challenge of Performance Assessment

As discussed in Chapter 2, the use of standard evaluation measures leads to misleading conclusions in the context of non-uniform utility predictive tasks. The metrics for performance evaluation must be focused on the utility achieved by the models rather than on their accuracy. This challenge has been solved with the proposal of several measures that allow to adequately evaluate the utility of the models (see Table 2.8 in Chapter 2, page 23). Still, we must highlight that, most of the proposed metrics for utility-based regression tasks are based on asymmetric loss functions that are not able to fully capture the concept of utility of the individual predictions. The work of Ribeiro [2011] has provided solutions for more general utility settings for regression tasks. Using the method proposed by Ribeiro [2011], it is possible to derive the MU (cf. Equation 2.19, page 24).

Ribeiro [2011] also proposed the NMU (cf. Equation 3.12), which is a normalised version of the MU measure that uses the utility-based regression framework described and represented by $U^p_\phi(\hat{y}_i, y_i)$. The NMU metric is calculated as follows:

$$NMU = \frac{\sum_{i=1}^{N} U^p_\phi(\hat{y}_i, y_i) + N}{2N} \tag{3.12}$$

where $U^p_\phi(\hat{y}_i, y_i)$ represents the utility of predicting $\hat{y}_i$ for a true value of y_i derived through the automatic method described in Ribeiro [2011] (cf. Equation 3.11 defined in page 82).

3.4 Imbalanced Domains Learning: Definition and Main Challenges

In this section we discuss the problem of learning from imbalanced domains presenting its main challenges and providing solutions for addressing them.

Before discussing the open challenges of learning in imbalanced domains, we will first discuss the problem definition, explaining its extension to regression tasks.

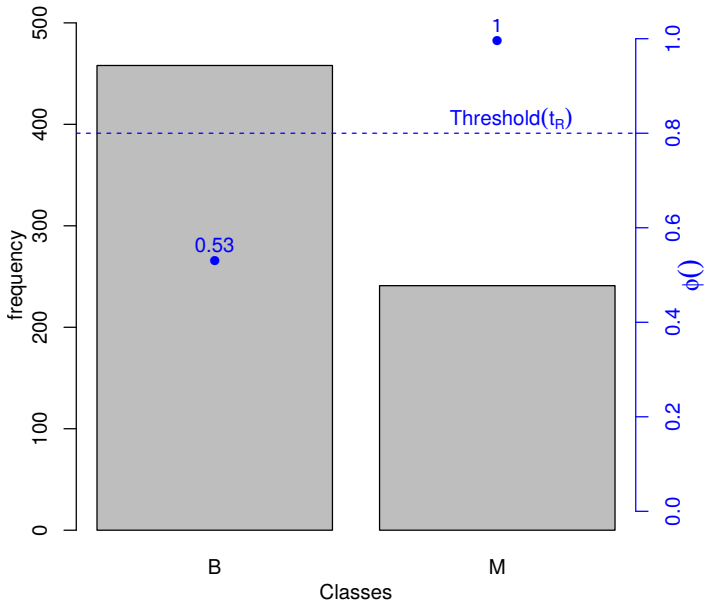


Figure 3.12: Probability mass function of a binary classification problem, relevance function ($\phi(Y)$) and relevance threshold (t_R).

3.4.1 The Problem of Learning from Imbalanced Domains

The problem of learning from imbalanced domains is a particular subclass of predictive tasks, more precisely, it is a subclass of utility-based learning problems. A definition for the imbalanced domain learning tasks has been presented in Section 3.2 (cf. Definition 3.2.11 in page 72). We face an imbalanced domain problem when the second component of the learning context θ' is a non-uniform relevance function and the relevance is higher on under-represented values of the target variable. Our goal is to analyse the provided definition of imbalanced domains, discussing its application for both classification and regression tasks. The notion of relevance function helps in defining which are the most important cases. However, to clearly separate the important from the non-important cases, we also need to define a threshold on the relevance function values. This threshold, t_R, sets the boundary above which the target variable values are relevant. Using t_R we are able to define a partition of the target variable domain $\mathcal{Y}$ in two complementary subsets: $\mathcal{Y}_R = \{y \in \mathcal{Y} : \phi(y) > t_R\}$ and $\mathcal{Y}_N = \mathcal{Y} \setminus \mathcal{Y}_R$. The subset $\mathcal{D}_R \subset \mathcal{D}$ contains the examples satisfying $y \in \mathcal{Y}_R$ and $\mathcal{D}_N = \mathcal{D} \setminus \mathcal{D}_R$. We must highlight that this partition is easily applicable to all regression problems as well as to binary and multiclass problems.

Figures 3.12 and 3.13 show the construction of sets $\mathcal{D}_R$ and $\mathcal{D}_N$ using the threshold (t_R) on the relevance values, for a classification and a regression problem, respectively.

Using this notation, the assertions describing the problem of learning from imbalanced domains (cf. Definition 3.2.11 provided in page 72) can be rewritten as follows:

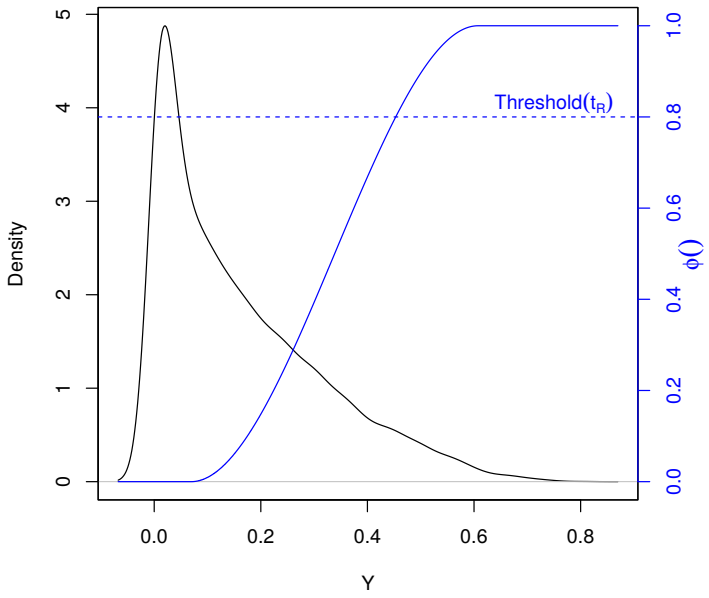


Figure 3.13: Probability density estimation, relevance function ($\phi(Y)$) and user defined threshold (t_R) for a regression problem.

i) $\theta' = \langle \mathcal{D}, \phi(Y) \rangle$, with $\phi(Y)$ non-uniform; and

ii) $|\mathcal{D}_R| << |\mathcal{D}_N|$.

The violation of any of the previous assertions results in the elimination of the problem of imbalanced domains. If assertion i) is not fulfilled, then all cases are equally relevant to the user, i.e., the user has a uniform interest over the domain ($\phi(Y)$ is uniform). In this case, this poses no problem to the learning procedure. If assertion ii) fails, this means that either the rare and normal cases distribution is approximately balanced, i.e., $|\mathcal{D}_R| \approx |\mathcal{D}_N|$ or the number of rare cases is much larger than the number of normal cases, i.e., $|\mathcal{D}_R| >> |\mathcal{D}_N|$. Both situations will also not represent a problem for the learner. In effect, in the first case, by having approximately the same number of normal and rare cases the learner will be able to focus on both type of cases. In the second case, by having available more rare cases the learner will be focused on these frequent cases which matches the user preferences are biased.

The problem of learning from imbalanced domains has been thoroughly studied in the context of classification tasks. However, for regression tasks, this has been a less explored issue. We address in the following section the main open challenges of the problem of learning from imbalanced domains for regression tasks. The main challenges that we will discuss are : i) how can the relevance function be obtained; ii) how can we evaluate the performance in these tasks in an adequate way; and iii) how can we develop models suitable for imbalanced domain tasks. We will discuss the two first challenges. The last challenge is addressed in Chapter 5 where several solutions are presented and evaluated.

3.4.2 The Challenge of Relevance Function Estimation

The formalisation of the domain knowledge provided through the user information tuple θ is one the biggest challenges that the data analyst must solve. Solutions for obtaining the utility surface information were presented in the previous Section 3.3.

As we have mentioned before imbalanced domains tasks are a special case of the more general utility-based learning task. The specificity lies on the fact that the end-user declares a preference for accurate performance on a subset of the values of the target variable domain (and as we have seen these values are rare). This information does not require a full utility matrix or surface, it is enough to say which are the relevant values. As we have seen before, that is the purpose of the relevance function - to map the target variable domain into a scale of relevance. In this context, the main goal of the formalisation of the user preferences in a imbalanced domains task is to be able to fully specify this relevance function.

Several methods can be used for obtaining an estimate of the relevance function of a given problem domain. The used method depends on the amount of available information provided by the end user. We will consider four different classes of information to express the user preferences in these problems:

- **Informal:** characterised by completely informal domain knowledge. This is typical in imbalanced domains where no quantification regarding the importance of each class or range of the target variable exist. Frequently, in classification tasks it is only stated that "the most important class is the minority".
- **Intermediate informal:** more information available although very limited. We assume the user provides a partial order of the classes or ranges of the target variable by their importance.
- **Intermediate formal:** characterised by a more complete information available. In this setting the user is able to provide a total order of the classes or ranges of the target variable.
- **Formal:** the user provides a full specification of the relevance function. Although being the ideal setting, this scenario is not common in real world domains.

Considering these classes of available information, different methods can be used to obtain an estimate of the relevance function. Obviously, in the "formal" class nothing is necessary as the user already specifies/provides the function.

The informal type of information is the most challenging one due to the lack of information. In this case the most suitable method involves the estimation of the relevance function based on the target variable distribution, because one of the properties of imbalanced tasks is the fact that the more important values are less frequent in the data set. More precisely, when

no information is provided, we can derive a relevance function that is inversely proportional to the target variable distribution. This is a valid method for classification and regression tasks. Following this intuition, in classification tasks, the relevance score assigned to each class i can be determined as:

$$\phi(i) = \frac{1/t_i}{\sum_{i=1}^{C} 1/t_i} \tag{3.13}$$

where i is a certain problem class, C is the total number of classes in the domain, and t_i is the sampling frequency of class i.

For regression tasks, Ribeiro [2011] proposed a similar method that allows to obtain automatically the relevance function. This method was already described in the previous Section 3.3 (in page 82) and uses statistics of the target variable distribution to obtain $\phi(Y)$.

When the available information is intermediate informal the end-user is already able to provide more information, although being still incomplete. In this case, the end-user is able to specify a partial order between the problem classes or ranges.

A partial order specifies a binary relation between pairs of target variable classes or ranges, depending if the problem is a classification or regression task. This relation is denoted as $c_1 < c_2$ and is read as "c_1 is the successor of c_2", where c_i represents a problem class or range. The relation $c_1 < c_2$ represents that c_1 has a lower relevance value than c_2. In the relation $c_1 < c_2$, c_1 is the successor of c_2 or, equivalently, c_2 is the predecessor of c_1. The relation is named partial because it does not provide a full relation between all the classes/ranges, i.e., there are pairs of classes/ranges named incomparable because the relation between both was not specified. Figure 3.14 shows on the left side an example of a partial order on a classification problem with 7 classes. In this figure we represent the binary relation $c_1 < c_2$ by a directed graph connecting nodes c_1 and c_2 with the arrow pointing to the predecessor, i.e. an arrow from c_1 (the successor) to c_2 (the predecessor). Several studies have been conducted to estimate rankings from a partial order (e.g. Brüggemann et al. [2004]). We propose the use the partial order of classes or ranges to estimate their relevance. The main advantage of this method is that it is less demanding for the end-user when compared to a full specification of the relevance function. Moreover, to use a partial order defined for the target variable is preferable to the situation where only informal information is available.

To estimate the relevance function for the classes (or ranges) of a certain problem using a partial order we applied the US-model [Brüggemann et al., 2004]. This method builds a Local Partial Order Model (LPOM) for each class/range. A LPOM for a node X represents all the successor (S), predecessor (P) and incomparable (U) nodes in relation to node X. Then, the estimated average rank of node X is defined as follows:

$$Rank(X) = \frac{(|S|+1) + (|S|+1+|U|)}{2} = |S| + 1 + \frac{|U|}{2} \tag{3.14}$$

Figure 3.14 shows, on the right side, the LPOM corresponding to node E. In this example

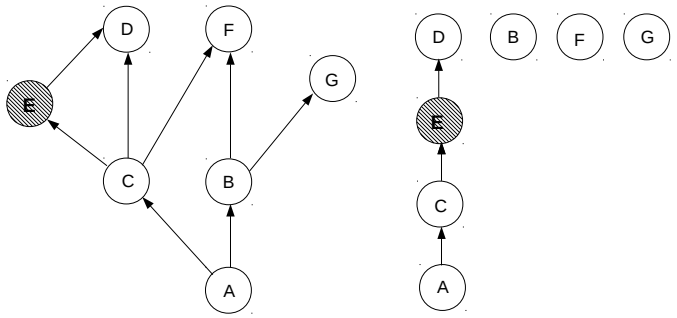


Figure 3.14: Example of a partially ordered set (left hand side) and the construction of a LPOM for class E (right hand side).

node E has 2 successors (nodes A and C), 1 predecessor (node D) and 3 incomparable nodes (B, F and G). Node E ranking, according to the proposed US-model, is $Rank(E) = 4.5$.

For classification tasks, we use the classes ranks derived from the partial order provided by the user and estimate the relevance of each class i as follows:

$$\phi(i) = \frac{Rank(i)}{\max_{\forall i \in C} Rank(i)} \tag{3.15}$$

For regression tasks a similar methodology can be applied as follows: for each user provided range of the domain i we use Equation 3.15 to derive the average value of $\phi(Y)$ for that range and then use an interpolation method to obtain the full continuous relevance function.

The last setting, regards the existence of intermediate formal information. In this case, we assume that the user is able to specify a total order of the problem classes or ranges and we use a mechanism similar to the previous one. This is a more demanding task for the user because all pairs of classes/ranges must be comparable, i.e., for each node considered there can not exist incomparable nodes. Given a total order, only the magnitude of the classes/ranges relevance remains unspecified. In this case, we can also use the US-model [Brüggemann et al., 2004] to obtain each class/range rank. For a given node X the formula applied to derive the corresponding rank is as follows: $Rank(X) = |S| + 1$. In the total order case we have a simplified formula because X has no incomparable nodes. After obtaining the rank of each class or range, the relevance can be estimated through Equation 3.15 and by applying the previously described mechanism for classification and regression problems.

3.4.3 The Challenge of Performance Assessment

As we have mentioned in Section 2.2, in the context of utility-based learning and, in particular, imbalanced domains learning, performance assessment is a critical issue. In these problems, the performance of a model should not be considered "blindly" with respect to the

errors location and must also take into account the error magnitude [Ribeiro, 2011]. Standard evaluation measures only take into account the magnitude of the errors independently of where they occurred. For this reason they are not suitable for these applications because they can be misleading [Ribeiro, 2011, Branco et al., 2016b].

The problem of performance assessment was thoroughly addressed within imbalanced classification problems and a diverse set of metrics was put forward as described in Chapter 2. The more frequently used metrics involve the aggregation into a single number of a given perspective derived from the problem confusion matrix. Because different metrics express different evaluation perspectives, it is usual to report the results of several metrics, to show the behaviour of the solutions across different angles. The most frequently used metrics for the class imbalance problem are the F_β (cf. Equation 2.27, in page 38), G-Mean (cf. Equation 2.28 in page 38). The results of AUC-ROC measure (cf. Equation 2.37 in page 40) are also frequently reported, although they have been shown to be misleading because they can provide an excessive optimistic view of the performance achieved [Davis and Goadrich, 2006].

When dealing with imbalanced regression problems only few solutions exist for assessing the performance. Generalisations of precision and recall measures, $prec^\phi$ (cf. Equation 2.38) and rec^ϕ (cf. Equation 2.39) respectively, have been proposed and were described in Chapter 2 (page 44). These two metrics can be aggregated into the F_β^ϕ metric using the same formula defined for classification (Equation 2.27). Important regression metrics proposed by Ribeiro [2011] are $AUC-ROC^\phi$, $AUC-PR^\phi$, $AUC-ROCIV^\phi$ and $AUC-PRIV^\phi$. These metrics were described in Section 2.4.2.2 (page 44).

The G-Mean is a metric often used in classification because it also takes into account the precision obtained on the negative class. In order to generalise G-Mean metric to a regression context we need to use the extension to regression problems of recall and specificity. The former metric has been proposed for regression as rec^ϕ (defined in page 44). We propose the generalisation to regression tasks of the specificity metric, which we call $spec^\phi$. This metric is focused on the performance achieved on the negative class and can be thought as the equivalent of precision measure but evaluated on the negative class.

The generalisation of specificity is obtained through a method similar to the one proposed for the obtaining precision and recall for regression. We use the concepts of utility (u) and relevance (ϕ) of a pair of true and predicted values of the target variable ($y_i, \hat{y}_i$). $spec^\phi$ is the proportion between the utility achieved by the model on the true normal cases ($\phi(y) \leq t_R$) and the maximal achievable utility (i.e., the relevance) for those cases as defined in Equation 3.16. The constant 1 is used in this equation in order to obtain measures in the interval $[0, 1]$.

$$true\ negative\ rate^{\phi}\ (specificity^{\phi}\ \text{or}\ spec^{\phi}):\ TN^{\phi}_{rate} = \frac{\sum_{\phi(y_i)\leq t_R}(1+u(\hat{y}_i,y_i))}{\sum_{\phi(y_i)\leq t_R}(1+\phi(y_i))} \quad (3.16)$$

Using these metrics we are able to adapt to regression the G-Mean [Kubat et al., 1998] defined for classification tasks. For distinguishing between the two evaluation measures we will refer to the regression version of the G-Mean, as $G-Mean^{\phi}$ because it involves the notion of relevance.

$$\begin{aligned} G-Mean^{\phi} &= \sqrt{rec^{\phi}\cdot spec^{\phi}} \\ &= \sqrt{\frac{\sum_{\phi(y_i)>t_R}(1+u(\hat{y}_i,y_i))}{\sum_{\phi(y_i)>t_R}(1+\phi(y_i))}\cdot\frac{\sum_{\phi(y_i)\leq t_R}(1+u(\hat{y}_i,y_i))}{\sum_{\phi(y_i)\leq t_R}(1+\phi(y_i))}} \end{aligned} \quad (3.17)$$

One advantage of $G-Mean^{\phi}$ is that it considers the accuracy achieved both on the rare and normal cases, i.e., it takes into account the performance of the cases with $\phi(y) > t_R$ and $\phi(y) \leq t_R$. On the other hand, the F_β measure is solely focused on the rare cases, integrating the accuracy achieved on both predicted and true rare cases.

For completeness, we also propose the extension of the negative predictive value to regression, which is the equivalent of rec^{ϕ} evaluated on the normal cases. The negative predictive value (NP^{ϕ}_{value}) is a proportion between the utility achieved on the predicted normal cases and the maximal achievable utility for those cases as defined in Equation 3.18. We use the constant 1 in the equation to obtain measures in the interval $[0,1]$.

$$negative\ predictive\ value^{\phi}:\ NP^{\phi}_{value} = \frac{\sum_{\phi(\hat{y}_i)\leq t_R}(1+u(\hat{y}_i,y_i))}{\sum_{\phi(\hat{y}_i)\leq t_R}(1+\phi(\hat{y}_i))} \quad (3.18)$$

In the following Example 3.4.1 we show how standard evaluation metrics can be misleading in the context of imbalanced regression problems and show that the use of different evaluation metrics may lead to different conclusions.

Example 3.4.1 (Evaluating the Air Quality Prediction Problem) *Let us consider once again the LNO2 data set described in Example 3.3.1. High values of target variable LNO2 indicate a bad air quality as opposed to lower LNO2 values. However, both extremes (low and high) are rare in the data set. Figure 3.15 shows the boxplot and the density function of the target variable approximated through a kernel density estimator.*

Let us suppose that a decision maker is interested in predicting the LNO2 variable for determining when to impose traffic restrictions to prevent reaching a dangerous atmosphere. In this case, the decision maker's preferences are not uniform across the target variable

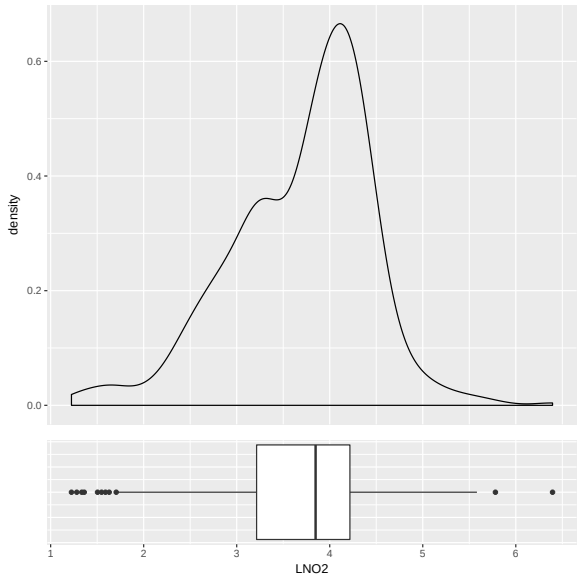


Figure 3.15: Target variable distribution of LNO2Emissions data set.

domain, and his main goal is to obtain a predictive model with high accuracy on high extreme values of LNO2. Figure 3.16 displays a relevance function suitable for these goals that was obtained using the automatic method proposed by Ribeiro [2011].

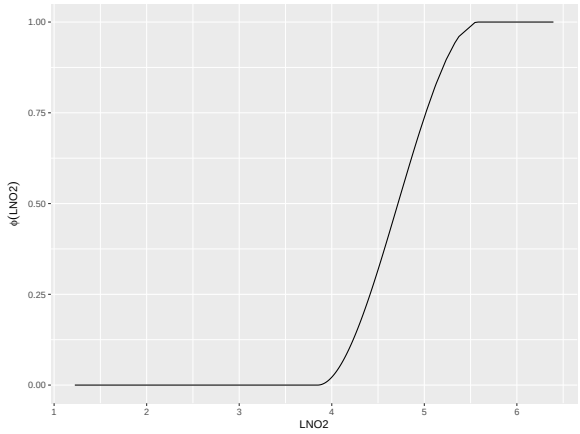


Figure 3.16: Relevance function automatically estimated for the target variable LNO2.

Let us consider a test sample with 10 examples of this data set. Figure 3.17 shows the predictions obtained by four artificially generated models. The relevance threshold (t_R) was set to 0.8, which in this case means that relevant target values are greater or equal to approximately 5.08.

By a simple observation of Figure 3.17 we notice that model $m1$ is the best model for predicting the higher LNO2 values. However, this model also signals several normal events as relevant, presenting generally inflated predictions. Model $m2$ comes next in what concerns

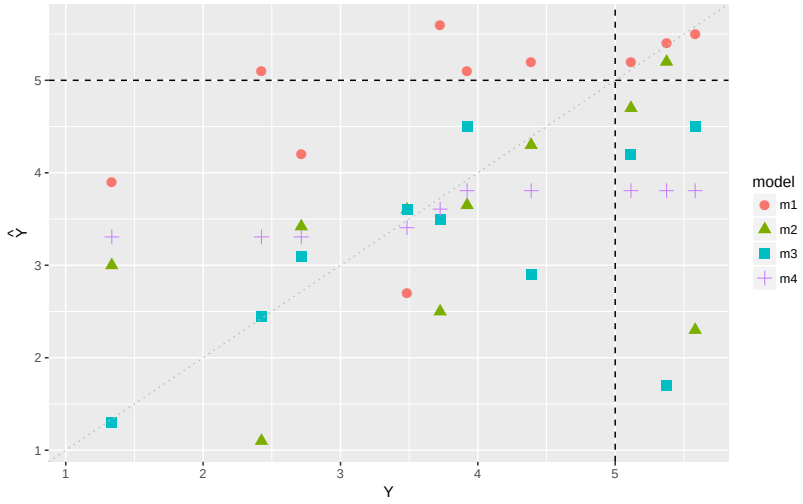


Figure 3.17: Predictions of four artificial models on a test sample with 10 cases drawn from the LNO2Emissions data set.

the rare case predictions, while the remaining models show a poor performance on these cases. Table 3.13 shows the results obtained by these four models on different metrics. We have grouped the measures into four groups defined as follows: i) standard regression metrics; ii) confusion matrix derived regression metrics; iii) metrics that aggregate precision/recall metrics (focused on rare cases performance only); and iv) metrics that aggregate the information of measures focused on rare and normal cases.

Standard measures such as MAE or MSE signal either model $m3$ *or model* $m4$ *as the best performing, respectively. When considering these metrics, the worst model is* $m1$*. However, provided that the higher values are the most important for the user, these conclusions are clearly misleading. This illustrates that, when tackling imbalanced domains learning problems, it is necessary to use suitable evaluation measures.*

The results of $prec^{\phi}$ *and* rec^{ϕ} *point models* $m1$ *or* $m2$ *as the best performing ones. This is in accordance with the expected because these metrics are focused on the rare values evaluation. On the other hand, both* $spec^{\phi}$ *and* NP^{ϕ}_{value} *position model* $m1$ *in last, although they do not agree on the best model. This is also the expected result because these measures are concentrated in the performance achieved in the normal cases.*

The next group of measures presented in Table 3.13 is based on aggregating the $prec^{\phi}$ *and* rec^{ϕ} *metrics. In imbalanced classification tasks, the* F_1 *measure (corresponding to the* F^{ϕ}_1 *for regression) is frequently reported. The results of this group of metrics point to either* $m1$ *or* $m2$ *as the best performing models. Still, we must highlight that the results of these metrics do not agree completely, although having the same overall focus.*

Finally, the last group of measures include metrics that aggregate information related with both rare and normal values. These metrics unanimously rank first model $m1$ *(and* $m3$ *in*

Table 3.13: Different metrics results on the four artificial models displayed in Fig. 3.17.

	Scores				**Decision Rank**			
	m1	*m2*	*m3*	*m4*	*m1*	m2	m3	*m4*
MAE	1.157	0.927	**0.852**	0.899	4	3	1	2
MSE	2.213	1.76	1.827	**1.271**	4	2	3	1
$prec^{\phi}$	0.695	**1.000**	0.000	0.000	2	1	3.5	3.5
rec^{ϕ}	**0.988**	0.735	0.640	0.674	1	2	4	3
$spec^{\phi}$	0.904	**0.998**	0.968	0.988	4	1	3	2
NP^{ϕ}_{value}	0.96	0.994	0.997	**1.000**	4	3	2	1
F_1^{ϕ}	0.816	**0.847**	0.000	0.000	2	1	3.5	3.5
$AUC-PR^{\phi}$	**1.000**	0.826	0.826	**1.000**	1.5	3.5	3.5	1.5
$AUC-PRIV^{\phi}$	**0.995**	0.813	0.763	0.847	1	3	4	2
$AUC-ROC^{\phi}$	**0.944**	0.5	**0.944**	0.888	1.5	4	1.5	3
$AUC-ROCIV^{\phi}$	**0.990**	0.499	0.882	0.77	1	4	2	3
$G-Mean^{\phi}$	**0.945**	0.856	0.787	0.816	1	2	4	3

the case of $AUC-ROC^{\phi}$). However, the remaining ranks of the models display differences.

This example shows that standard metrics should not be used in the context of imbalanced regression problems. It also demonstrates that different evaluation measures capture different perspectives of the evaluation. Moreover, it is also shown that the metrics may provide different results, even when considering metrics that aggregate the same base measures. This means that, even when suitable metrics are used, the evaluation results may change according to the metric being observed.

3.5 Conclusions

In this chapter we have proposed a utility-based learning framework. This framework involved the formalisation of predictive tasks, which allowed to establish important connections between different types of predictive tasks. In particular, the proposed framework allowed to verify that standard learning tasks are a special case of utility-based learning problems. We have also seen that the problem of learning from imbalanced domains can be seen as a special case of utility-based learning tasks.

An important set of open issues in utility-based regression tasks was presented. We discussed these open challenges and presented solutions for them for both utility-based regression and imbalanced domains learning tasks.

The two following chapters focus on modelling solutions for both utility-based regression problems and imbalanced domains learning tasks.

Chapter 4

Utility Optimisation for Regression Tasks

If we have information on the utility preferences of the end-user we should use it to drive the learning of predictive models. In this chapter we address the problem of maximising utility in regression tasks. We propose two solutions to the problem of maximising utility and evaluate them on several real world tasks.

4.1 Introduction

Many important practical applications involve the consideration of a utility setting. In the previous chapter we defined a framework for the problem of utility-based learning, and showed its relations with other predictive tasks. We also discussed the main challenges involving utility-based learning problems, particularly concerning the procedures used to express the user preferences in a way that is useful for learning predictive models.

In this chapter we focus on the problem of utility-based regression, assuming we have an utility surface that describes the user preference biases. Given a utility surface, how can we change the learning algorithms in order to obtain models that achieve higher utility scores? We answer this question in Sections 4.2 and 4.3 by proposing two solutions to optimise utility in regression tasks: UtilOptim and MetaUtil algorithms. In Section 4.4, we evaluate and discuss the predictive performance of the proposed algorithms through an extensive experimental study. Finally, in Section 4.5 we conclude this chapter.

4.2 UtilOptim: Maximising the Expected Utility

Our first proposal to obtain models biased towards utility maximisation is based on the estimation of the maximum expected utility for a given test case. The key idea of this approach is to determine the optimal prediction for a case by maximising its expected utility. To calculate the expected utility we use the conditional density of the target variable. Let $f_{Y|X}$ represent the conditional probability density function of Y given the occurrence of the value $\mathbf{x}$ of X.

For a given test case $q = \langle \mathbf{x}_k, y_k \rangle$, with y_k unknown, the optimal prediction y^* that maximises the expected utility, can be determined as follows:

$$y^* = \arg\max_{z \in Y} \int f_{Y|X}(y|X = \mathbf{x}_k) \cdot U(z, y)\ dy \tag{4.1}$$

Equation 4.1 shows how the optimal prediction for a given example can be obtained, assuming the true conditional density for the target variable is known. This equation is the extension to regression and utility-based problems of the minimisation of the conditional risk [Duda et al., 2012] described in Equation 2.8 (page 13, Section 2.3.1). To apply the described mechanism in practice, we need: i) a utility surface describing the user preference bias; ii) an estimate of the conditional probability density function $f_{Y|X}$; and iii) an estimate for the integral in Equation 4.1. Regarding the definition of the utility surface, we have described several approaches in the previous chapter. To obtain an estimate for $f_{Y|X}$, we propose the use of the method presented by Frank and Bouckaert [2009] and Rau et al. [2015] that uses a class probability estimator to approximate $f_{Y|X}$ in a certain range $[y_l, y_h]$ of the target variable Y. Finally, we can use the trapezoidal rule (e.g. Davis and Rabinowitz [2007]) to approximate the integral in the equation through a definite integral in $[y_l, y_h]$. Algorithm 4.1 describes with more detail the process used to estimate the optimal prediction under a utility-based regression setting.

The method proposed by Frank and Bouckaert [2009] and Rau et al. [2015] is used to obtain an estimate of $f_{Y|X}$. This method is based on the usage of a class probability estimator $\hat{p}$ that is able to provide the probabilities of each class c conditioned on a given test case q, i.e. $\hat{p}(c|q)$. Such estimator can easily be obtained using some training set of the problem. We create such training set by using the available data of the regression task, but changing the continuous target to a discretized version, i.e. a nominal variable. This results from discretizing the continuous target variable values into a set of B contiguous, non-overlapping and equal width bins. The considered bins cover the interval $[y_l, y_h]$ of the target variable Y. Each of these bins will act as a class value. After training the probabilistic classifier with this data, the idea is to use the estimated class probabilities, $\hat{p}(c|q)$, to derive weights for each bin. Given a case $\mathbf{x}_q$, we derive a weight w_b for each bin $b \in B$ as follows:

Algorithm 4.1 Utility Optimization (UtilOptim).

Input: $Train$ - regression training set with target variable Y
U - utility surface (Optional)
A - probabilistic classifier
ϵ - granularity
$Test$ - test set

Output: Predictions obtained for $Test$

1: **function** UtilOptim($Train$, U, A, ϵ, $Test$)
2: **if** U is not defined **then**
3: $U \leftarrow$ estimate utility using the automatic method provided by Ribeiro [2011]
4: **end if**
5: $B \leftarrow$ set of ϵ-width non-overlapping bins obtained by discretizing Y in $Train$
6: $V \leftarrow$ set of mid-points of each bin b
7: **for each** $\mathbf{x}_t \in Test$ **do**
8: **for each** $b \in B$ **do**
9: $w_b(\mathbf{x}_t) \leftarrow \dfrac{\hat{p}(b|\mathbf{x}_t)}{|b|}$ ▷ Use A and $Train$ to estimate $\hat{p}(b|\mathbf{x}_t)$
10: **end for**
11: **for each** $v \in V$ **do**
12: $f_{Y|X}(v|\mathbf{x}_t) \leftarrow \sum_{i=1}^{n} w_{b:y_i \in b}(\mathbf{x}_t) K_h(v - y_i)$
13: **end for**
14: $\hat{y}_t \leftarrow \arg\max_{z \in Y} \int f_{Y|X}(v|\mathbf{x}_t) \cdot U(z, v)\ dv$ approximated through a trapezoidal rule with granularity ϵ
15: **end for**
16: **return** $\hat{y}_t\ \forall \mathbf{x}_t \in Test$
17: **end function**

$$w_b(\mathbf{x}_q) = \frac{\hat{p}(b|\mathbf{x}_q)}{|b|} \tag{4.2}$$

where $|b|$ is the number of training cases in the bin b, and $\hat{p}(b|\mathbf{x}_q)$ is the estimated probability of the bin b given the test case $\mathbf{x}_q$.

We then use a weighted Gaussian kernel density estimator with bandwidth h in conjunction with the obtained weights to yield the following estimate of the conditional probability density function:

$$f_{Y|X}(y|X = \mathbf{x}_q) = \sum_{i=1}^{n} w_{b:y_i \in b}(\mathbf{x}_q) K_h(y - y_i) \tag{4.3}$$

where $w_{b:y_i \in b}$ represents the weight of the bin to which y_i belongs, $K_h(x) = \frac{K(x/h)}{h}$ represents a Gaussian kernel, i.e., $K(x) = \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{x^2}{2}}$, and h is the kernel bandwidth. The bandwidth value determines how closely the estimator adjusts to the data. The value that we selected is based on the global standard deviation and the number of data points. We followed the recommendation of Silverman [1986] for setting the h value, namely we used $h = 0.9An^{-1/5}$, where $A = min\{\sigma_X, \frac{IQR}{1.34}\}$, IQR represents the interquartile range and σ_X is the standard deviation, both estimated with the training data.

Finally, to calculate the optimal prediction of a case $\mathbf{x}_q$ it is necessary to evaluate the integral of Equation 4.1. This integral is approximated by a definite integral in the interval $[y_l, y_h]$ as shown in Equation 4.4. To do this, we need to assess the effects of predicting z for case $\mathbf{x}_q$, for all $z \in [y_l, y_h]$. The value of z that yields the highest expected utility is selected as the optimal prediction for $\mathbf{x}_q$. We use a parameter ϵ for setting the granularity to use. Given this granularity ϵ, we evaluate the function $f_{Y|X}$ and the surface U in $[y_l, y_h]$ using a set of points equality spaced by ϵ. We also use this set of points to estimate the value of the integral in Equation 4.4 using a trapezoidal rule (e.g. Davis and Rabinowitz [2007]).

$$y^* = \underset{z \in [y_l, y_h]}{\arg\max} \int_{y_l}^{y_h} f_{Y|X}(y|X = \mathbf{x}_k) \cdot U(z, y)\ dy \tag{4.4}$$

Figure 4.1 illustrates this procedure for a test case of the LNO2Emissions data set. In this data set the high extreme values of the target variable are more valuable for the end-user and they configure a situation with extreme pollution, which is a serious event with potential impact in human health. The solid black line represents the $f_{Y|X}$ estimate obtained conditional on the considered test case. The red dashed line displays the values obtained for the integral proposed in Equation 4.1 across the Y domain. The vertical lines in the figure show the true target variable value (dashed blue), the LNO2 value with the highest probability given the information in the probability density function conditional on the case (solid black) and the LNO2 value with the highest expected utility (dashed red), which

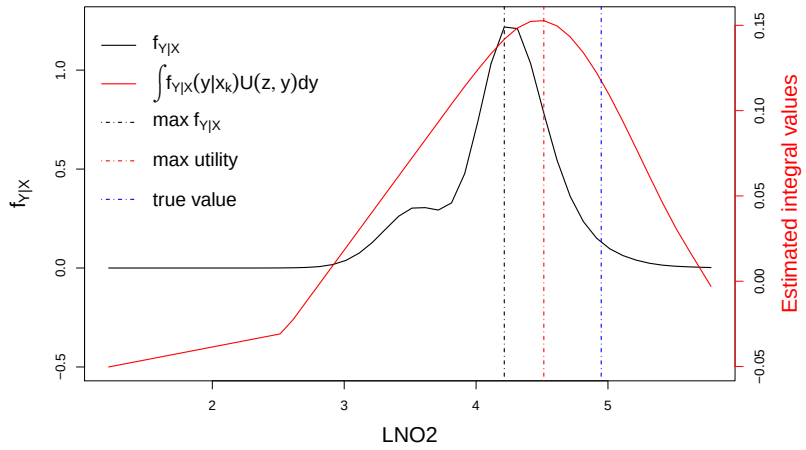


Figure 4.1: Results from utility optimisation and $f_{Y|X}$ estimation for one LNO2Emissions data example with true target variable value of 4.949.

will be the predicted value, in accordance with Equation 4.1. Note how, in this illustrative example, the prediction is pushed to a value with lower conditional probability but higher expected utility.

4.3 MetaUtil: Maximising the Utility by Changing the Training Set

Our second proposal, MetaUtil, is able to obtain a model that tries to maximise utility by changing the provided training set. This method is an adaptation of the idea proposed by Domingos [1999] and implemented in the algorithm MetaCost for classification tasks. In MetaCost the idea is to change the training data in such a way that when learning a model with the new data we obtain a model that minimises the cost of the predictions. Here, we adapt this idea to a context of maximising utility for regression tasks.

The key idea of the MetaUtil algorithm is to change the target variable value of each training case to the value we estimate that will maximise the expected utility. As MetaCost, the MetaUtil algorithm uses several bootstrap samples to obtain different models to provide conditional probability density estimates. Each of these models is used to obtain approximations of the conditional probability density function, $f_{Y|X}(y|X = \mathbf{x})$, using the same procedure described in the UtilOptim algorithm. There are two different ways for obtaining the final estimate of $f_{Y|X}(y|X = \mathbf{x})$: i) by averaging all the approximations provided by these models; or ii) by averaging only the approximations obtained from samples that did not include the example $\mathbf{x}$. This final average estimate is used in Equation 4.1 to derive the optimal y value for a given case. As in the MetaCost algorithm, the original target

variable value of the each training case is replaced by this optimal y value. Finally, a model is trained with this modified training set. In summary, MetaUtil uses a bootstrap procedure to obtain the optimal target variable value for each training case, according to the preference biases of the user. In MetaCost these preferences are expressed through a cost matrix, and the optimal target values calculated using the conditional risk (Equation 2.8). In MetaUtil a utility surface describes the user preferences and the optimal values are calculated using Equation 4.1. MetaUtil obtains a new, modified training set, where the target variable values were changed in accordance with the user preference biases, i.e., the new target variable values maximise the expected utility. This new training set is used to obtain the final model that is expected to be biased towards the user preferences as it was learned in a training set modified with the utility information that describes these preferences. The details of our proposal are described in Algorithm 4.2.

The MetaUtil algorithm has the important advantage of allowing to obtain more interpretable models. This happens because the modifications are embedded in the new training set, and therefore, the learned model is obtained with a training set biased towards the utility preferences of the user. This means that this model will reflect the user preferences as expressed through the utility surface.

4.4 Experimental Analysis

In this section we describe and discuss the results of an experimental evaluation of the two approaches for utility optimisation. Our main goal is to assess the effectiveness of using the UtilOptim and the MetaUtil algorithms presented in Sections 4.2 and 4.3. More specifically, we carried out a set of experiments for evaluating: i) the advantages of using the proposed approaches; and ii) the impact in the predictive performance when applying these approaches under different utility settings.

4.4.1 Materials and Methods

Data Sets and Utility Settings. In the experiments carried out we used 14 real world regression data sets from different domains. For each data set, as we had no access to domain experts, we have used the automatic method proposed by Ribeiro [2011] to derive a relevance function and then a utility surface. Table 4.1 shows the main characteristics of these data sets, including the number and percentage of rare values for a relevance threshold (t_R) of 0.8.

One goal of our experiments regards assessing the impact of using the proposed methods under different utility settings. To obtain different utility surfaces for each data set we applied the method proposed by Ribeiro [2011]. This method allows to derive a set of utility

Algorithm 4.2 MetaUtil.

Input: $Train$ - regression training set with target variable Y
A - regression learning algorithm
U - utility surface (Optional)
m - number of samples to generate
n - number of examples in each sample
r - TRUE *iff* all samples are to be used for each example
ϵ - granularity parameter
$Test$ - test set

Output: Predictions obtained for $Test$

function MetaUtil($Train$, A, U, m, n, r, ϵ, $Test$)
  for $i = 1$ **to** m **do**
    $S_i \leftarrow$ bootstrap sample of $Train$ with size n
    $M_i \leftarrow \{\widetilde{f}_{Y|X}\}_{\mathbf{x} \in Train}$ family of functions estimated with granularity ϵ using S_i
  end for
  for each example $\langle \mathbf{x}, y \rangle \in Train$ **do** ▷ Change training set target values
    if r is TRUE **then**
      $M'(\mathbf{x}) \leftarrow$ average of $M_i(\mathbf{x})$
    else
      $M'(\mathbf{x}) \leftarrow$ average of models $M_i(\mathbf{x})$ where $\mathbf{x}$ was not used for training
    end if
    $y \leftarrow \arg\max_{z \in Y} \int M'(\mathbf{x}).U(y, z)\ dy$ approximated through trapezoidal rule with granularity ϵ
  end for
  $M \leftarrow$ model obtained from applying A to the new modified training set
  for each test case $t := \mathbf{x}_t$ **do**
    $\hat{y}_t \leftarrow predict(\mathbf{x}_t, M)$
  end for
return $\hat{y}_t\ \forall t \in Test$
end function

Table 4.1: Characteristics of the 14 used data sets. (N: Nr of cases; *tpred*: Nr of predictors; *p.nom*: Nr of nominal predictors; *p.num*: Nr numeric predictors; *nRare*: nr. cases with $\phi(Y) > 0.8$; $\%Rare$: $100 \times nRare/N$).

Data Set	N	tpred	p.nom	p.num	nRare	% Rare
servo	167	4	2	2	34	20.4
a6	198	11	3	8	33	16.7
Abalone	4177	8	1	7	679	16.3
a3	198	11	3	8	32	16.2
a4	198	11	3	8	31	15.7
a1	198	11	3	8	28	14.1
a7	198	11	3	8	27	13.6
boston	506	13	0	13	65	12.8
a2	198	11	3	8	22	11.1
a5	198	11	3	8	21	10.6
fuelCons	1764	37	12	25	164	9.3
bank8FM	4499	8	0	8	288	6.4
Accel	1732	14	3	11	89	5.1
airfoild	1503	5	0	5	62	4.1

Table 4.2: Regression algorithms, parameter values, and respective R packages.

Learner	Parameter Variants	R package
SVM	$cost = \{10, 150\}$ $gamma = \{0.01, 0.001\}$	**e1071** [Dimitriadou et al., 2011]
Random Forest	$mtry = \{5, 7\}$ $ntree = \{500, 750, 1500\}$	**randomForest** [Liaw and Wiener, 2002]

surfaces representing different penalisation contexts, for each data set. To achieve this, we used different values for parameter p that specifies which types of errors should be more/less penalised. More precisely, the values of p are used to assign more or less costs to two different situations: opportunity costs (a relevant case was predicted as non relevant) and false alarms (a non relevant case was predicted as relevant). In the method proposed by Ribeiro [2011], when $p > 0.5$ opportunity costs are considered more serious than false alarms, i.e., missing a relevant prediction is considered to have a higher cost than predicting a non relevant case as relevant. On the other hand, when $p < 0.5$ the reverse happens: opportunity costs are less penalised than false alarms. To assess the impact of using different utility settings, we obtained three different utility surfaces for each data set by considering $p \in \{0.2, 0.5, 0.8\}$.

Learning settings. In our evaluation we used 2 different learning algorithms with different parameter variants. The algorithms, set of used parameter variants and respective used R packages are described in Table 4.2.

We applied 10 learning approaches (6 Random Forest variants + 4 SVM variants) to each of the 42 problems (14 data sets $\times$ 3 utility surface settings) using both the original regression

learner without utility optimisation (Orig) and the proposed approaches for optimising the utility: UtilOptim (cf. Algorithm 4.1) and MetaUtil (cf. Algorithm 4.2).

We fixed the granularity parameter ϵ to 0.1. Regarding the required probabilistic classifier, we selected the classification learner most closely related to the regression algorithm being compared against. The motivation for this choice is related with the negative impact in the observed performance when there is a mismatch between the probability estimator and the used classifier [Domingos, 1997]. Moreover, we will assume, as done by Domingos [1999], that the user is able to select the regression scheme that best adapts to the task that is being considered. This selected scheme is then used for learning the regression model and its probabilistic classifier counterpart is used for obtaining the conditional probability density estimates. In the MetaUtil algorithm we set the number of bootstrap samples to generate (parameter m) to 20; the number of examples in each sample (parameter n) to the training set size and parameter r to TRUE meaning that all samples are used.

Evaluation Methodology. All alternatives were evaluated using the Normalized Mean Utility (NMU) metric defined in Equation 3.12 (page 88). When dealing with utility-based regression tasks it is necessary to use suitable evaluation metrics, as we previously discussed in Section 3.3. We selected a normalised measure for obtaining comparable results across different data sets. The NMU values were estimated by 2 repetitions of a stratified 10-fold cross validation process. This process was done using the infrastructure provided by R package `performanceEstimation` [Torgo, 2014].

In addition to reporting the NMU scores, we also assessed the statistical significance of the results through the non-parametric Friedman F-test together with a post-hoc Nemenyi test with a significance level of 95%.

4.4.2 Results and Discussion

The 10 learning variants were applied to the 42 regression problems (14 data sets using 3 different utility surface settings) using 3 strategies for utility optimization (Orig, UtilOptim and MetaUtil). Thus, we tested 1260 ($10 \times 42 \times 3$) combinations. Tables 4.3 to 4.5 show the mean NMU results of the variants of each learner, obtained for each utility setting i.e., when considering different values for parameter p in the generation of the utility surface. The best results by learning algorithm and data set are displayed in bold.

The advantage of the proposed algorithms is clear for all the evaluated utility contexts. We also observe that, for the tested SVM variants, the algorithms display a more consistent advantage. The advantages are also noticeable for the RF variants. However, in some data sets, the proposed algorithms were not able to improve on the RF performance, which does not happen with the SVM variants.

Overall, the NMU results of both proposed algorithms are rather competitive. The MetaUtil method achieves several times the best average performance, particularly for utility surfaces with higher values of p.

Table 4.3: Mean NMU results of the variants of each learner by data set for the value of parameter p on the utility surface set to 0.2. (Best results by leaner and data set are in bold.)

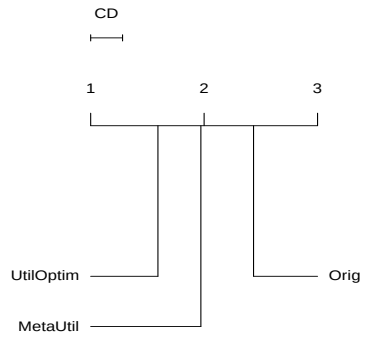
	SVM			RF		
	Orig	UtilOptim	MetaUtil	Orig	UtilOptim	MetaUtil
servo	0.4891	**0.5585**	0.4772	**0.5712**	0.5624	0.5666
a6	0.5157	**0.5223**	0.5203	0.5123	**0.5140**	0.5051
Abalone	0.5751	**0.5862**	0.5818	0.5786	0.5849	**0.5855**
a3	0.5039	**0.5108**	0.5085	0.5002	**0.5079**	0.4843
a4	0.5200	**0.5307**	0.5303	0.5269	0.5272	**0.5275**
a1	0.5335	**0.5371**	0.5370	0.5485	**0.5498**	0.5490
a7	0.4983	0.5075	**0.5083**	0.4735	**0.5055**	0.4543
boston	0.5748	**0.5770**	0.5738	0.5784	**0.5806**	0.5784
a2	0.5236	0.5294	**0.5312**	**0.5289**	0.5276	0.5260
a5	0.5226	0.5284	**0.5288**	**0.5269**	0.5250	0.5222
fuelCons	0.6138	**0.6183**	0.6164	0.6175	**0.6257**	0.6213
bank8FM	0.5694	0.5699	**0.5720**	0.5703	0.5690	**0.5707**
Accel	0.5582	**0.5606**	0.5593	0.5638	**0.5655**	0.5641
airfoild	0.4566	**0.4853**	**0.4853**	0.4599	**0.4853**	**0.4853**

Table 4.4: Mean NMU results of the variants of each learner by data set for the value of parameter p on the utility surface set to 0.5. (Best results by leaner and data set are in bold.)

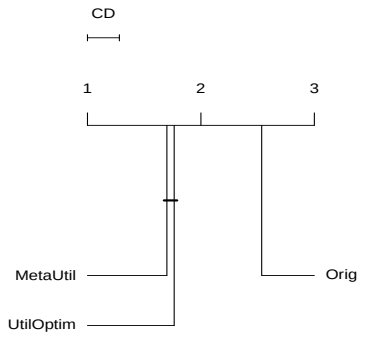
	SVM			RF		
	Orig	UtilOptim	MetaUtil	Orig	UtilOptim	MetaUtil
servo	0.4875	**0.5601**	0.4778	**0.5718**	0.5655	0.5660
a6	0.5072	**0.5207**	0.5187	**0.5140**	0.5069	0.5116
Abalone	0.5705	**0.5893**	0.5829	0.5764	**0.5884**	0.5883
a3	0.4927	**0.5051**	0.5046	**0.5048**	0.4958	0.4944
a4	0.5140	0.5313	**0.5336**	0.5284	0.5331	**0.5346**
a1	0.5297	0.5394	**0.5425**	0.5479	0.5531	**0.5533**
a7	0.4859	0.4975	**0.4976**	0.4810	**0.4840**	0.4652
boston	0.5743	**0.5775**	0.5748	0.5782	**0.5814**	0.5792
a2	0.5180	0.5289	**0.5309**	0.5269	0.5242	**0.5298**
a5	0.5172	0.5282	**0.5293**	0.5257	0.5261	**0.5269**
fuelCons	0.6135	**0.6194**	0.6170	0.6171	**0.6259**	0.6222
bank8FM	0.5692	0.5702	**0.5723**	0.5703	0.5694	**0.5710**
Accel	0.5580	**0.5611**	0.5601	0.5638	**0.5655**	0.5643
airfoild	0.4508	**0.4633**	**0.4633**	0.4575	**0.4635**	0.4631

Table 4.5: Mean NMU results of the variants of each learner by data set for the value of parameter p on the utility surface set to 0.8. (Best results by leaner and data set are in bold.)

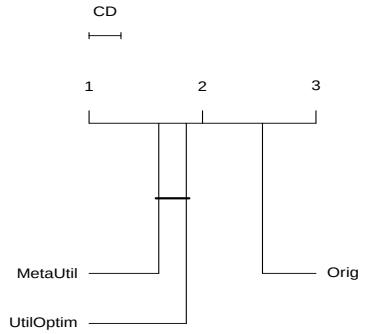
	SVM			RF		
	Orig	UtilOptim	MetaUtil	Orig	UtilOptim	MetaUtil
servo	0.4859	**0.5624**	0.4840	**0.5723**	0.5661	0.5664
a6	0.4987	0.5251	**0.5259**	0.5156	0.5128	**0.5239**
Abalone	0.5660	**0.5961**	0.5866	0.5743	**0.5955**	0.5939
a3	0.4814	0.5132	**0.5152**	**0.5093**	0.4960	0.5067
a4	0.5081	0.5412	**0.5481**	0.5299	0.5473	**0.5484**
a1	0.5258	0.5490	**0.5539**	0.5473	0.5589	**0.5612**
a7	0.4735	**0.4950**	0.4926	**0.4886**	0.4728	0.4793
boston	0.5737	**0.5783**	0.5762	0.5780	**0.5819**	0.5804
a2	0.5123	0.5330	**0.5352**	0.5249	0.5293	**0.5373**
a5	0.5119	0.5326	**0.5353**	0.5244	0.5312	**0.5353**
fuelCons	0.6131	**0.6207**	0.6175	0.6167	**0.6260**	0.6233
bank8FM	0.5691	0.5708	**0.5725**	0.5702	0.5699	**0.5713**
Accel	0.5577	**0.5618**	0.5612	0.5637	**0.5656**	0.5646
airfoild	0.4450	**0.4460**	0.4422	**0.4550**	0.4509	0.4509



(a) Results for Utility Surface with $p = 0.2$.



(b) Results for Utility Surface with $p = 0.5$.



(c) Results for Utility Surface with $p = 0.8$.

Figure 4.2: Critical Difference diagrams of average NMU results for different utility surface settings.

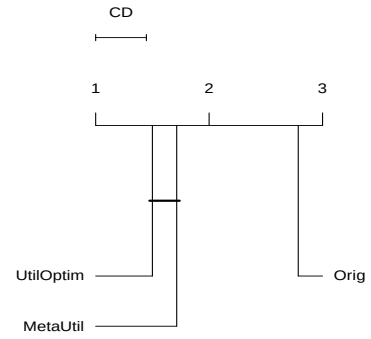
We proceeded with the application of the non-parametric Friedman F-test, to check the statistical significance of the observed differences. The F-test results allowed the rejection of the null hypothesis that all the tested approaches for utility optimisation exhibit the same performance. We then applied the post-hoc Nemenyi test with a significance level of 95% to verify which approaches are statistically different. The critical difference diagrams (CD diagrams) [Demšar, 2006] with the results aggregated by type of utility surface settings are displayed in Figure 4.2. The lower average ranks indicate a better performance and when two algorithms are connected by a bold horizontal line it means that their average ranks are not significantly different, i.e. the observed difference in performance is not statistically significant. When considering the performance of all tested learner variants we can conclude that: i) using the original learner has the worst performance for all tested utility settings with statistical significance; ii) the UtilOptim is better, with statistical significance, than the remaining tested approaches for the utility surfaces obtained with the lower value of p; iii) for the remaining utility surface settings, although the MetaUtil algorithm displays a lower rank, the observed differences between the UtilOptim and the MetaUtil algorithms are not statistically significant.

With the goal of providing a more detailed picture of the performance of the algorithms, we also present the CD diagrams corresponding to the three tested utility surface settings for the SVM and RF variants, separately. The results of the SVM variants are displayed in Figure 4.3, while those of RF variants are shown in Figure 4.4.

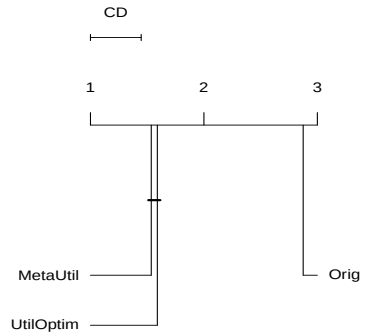
The SVM results confirm that for this learner there is no statistical significance between the performance of the UtilOptim and MetaUtil algorithms in all tested utility settings. The UtilOptim algorithm achieves a lower rank for the utility surface with lower value of parameter p, while the MetaUtil has a better rank for the more balanced utility setting ($p = 0.5$).

Regarding the RF results, the better performance of UtilOptim algorithm is also confirmed for the lower value of p in the tested utility surface settings. For the remaining values of p, the MetaUtil algorithm has a better performance as it provides lower ranks in the CD diagrams, although not always with statistical significance when compared against UtilOptim algorithm.

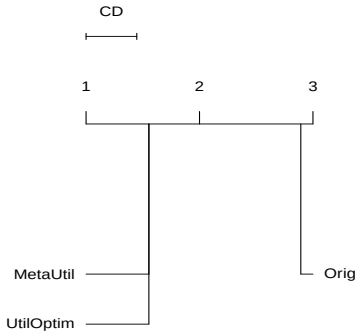
When observing the CD diagrams results by learner, we can conclude that: i) using the original learning algorithm is worst with statistical significance under all tested utility settings; ii) for the SVM learner, the differences between UtilOptim and MetaUtil algorithms are not statistically significant, although UtilOptim achieves a lower rank for the more extreme utility settings and MetaUtil has a lower rank on the remaining utility surfaces; iii) for the RF learner, UtilOptim is better with statistical significance for the lower value of p, while MetaUtil achieves a lower rank on the remaining utility surface settings, although only for the $p = 0.8$ this is statistically significant.



(a) Results for Utility Surface with $p = 0.2$.

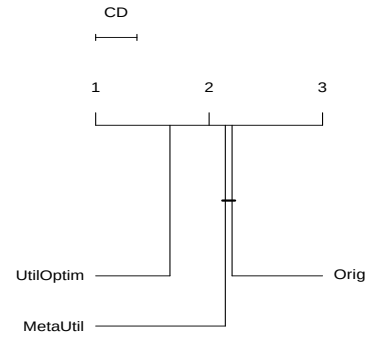


(b) Results for Utility Surface with $p = 0.5$.

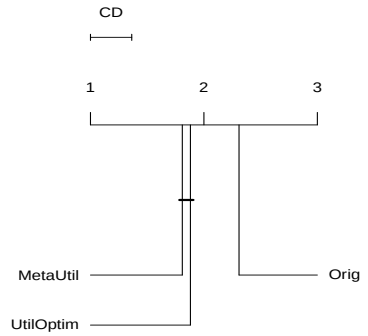


(c) Results for Utility Surface with $p = 0.8$.

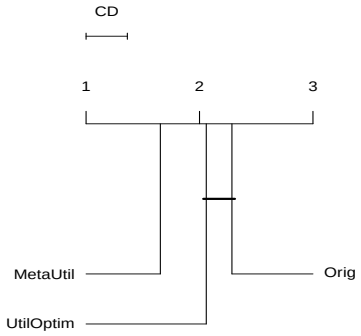
Figure 4.3: Critical Difference diagrams of average NMU results for SVM learner.



(a) Results for Utility Surface with $p = 0.2$.



(b) Results for Utility Surface with $p = 0.5$.



(c) Results for Utility Surface with $p = 0.8$.

Figure 4.4: Critical Difference diagrams of average NMU results for RF learner.

Based on the ranks described in Figures 4.3 and 4.4, we would generally recommend the use of UtilOptim when lower values of p are being used for setting the utility surface; and the MetaUtil algorithm for the remaining utility surface settings. We can say that the UtilOptim generates more conservative models that are less prone to "false alarms", while the MetaUtil algorithm generates more risky models that are less prone to "opportunity costs". Still, we must highlight that, in the majority of the situations there is no statistical significance between the results achieved by UtilOptim and MetaUtil algorithms.

In spite of the similar predictive performance of the two proposed methods, there are significant differences between them in terms of interpretability and computation costs. The interpretability factor favours the MetaUtil algorithm while the computation aspect favours the UtilOptim approach.

When using as base learners algorithms that produce interpretable models, the MetaUtil algorithm learns these models with a training set biased toward the utility preferences of the end user. This means that the models will reflect these biases and thus if the user wants a justification for some predicted value she/he will be able to find it in the model. On the other hand, UtilOptim works as a post-processing method, meaning that the models are learned without any consideration of the user utility preferences. It is only when making the predictions that the values predicted by the learned models may be changed towards new values that maximise the expected utility. However, the consequence is that if the user wants a justification for this predicted value she/he will not be able to find it in the models, because the models have not predicted that value, the value resulted from the applied change. In this sense, we can say that the MetaUtil algorithm produces models that are better from the interpretability perspective, provided the used base algorithms are interpretable.

From the perspective of computational complexity, the UtilOptim algorithm has a clear advantage as only one estimate of the conditional probability density function is obtained, while MetaUtil needs to obtain several of these estimates using different bootstrap samples.

4.5 Conclusions

The research community has proposed several different approaches for tackling cost-sensitive classification tasks and few attention has been given to the more general problem of dealing with utility information where both costs and benefits are considered. Moreover, the proposed solutions have been focused on classification tasks, existing few algorithms for tackling this problem in a regression context. The utility-based regression problem raises different challenges that must be considered. In this chapter we addressed the problem of learning regression models that try to maximise the utility of their predictions.

Learning models that maximise the utility according to a utility surface reflecting the user

preferences is an important challenge. We address this problem proposing two algorithms designed to maximise the models utility. An experimental analysis is carried out to assess the effectiveness of the proposed algorithms. The goals of the experimental analysis are two fold: i) to analyse the predictive performance of the proposed algorithms; and ii) to understand the impact of using different utility settings in the predictive performance of the algorithms. Concerning the first objective, the experimental evaluation carried out shows that both algorithms proposed have a clear advantage when compared against the original regression algorithm. It is also shown that it is difficult to determine which algorithm performs better. Among the possible factors that may influence the performance of UtilOptim and MetaUtil algorithms we find: the utility surface, the learning algorithm, or even data related characteristics. As for the second objective, results show that for all tested utility settings the observed differences between the two proposed algorithms were generally not statistically significant. By inspecting the rankings achieved by the algorithms, we recommend the use of UtilOptim algorithm when the utility surface is build with lower values of p. MetaUtil is recommended in the remaining situations. The end-user may also prefer the UtilOptim algorithm due to its lower computational cost or may select the MetaUtil algorithm if interpretability of the models is an important factor.

Chapter 5

Learning in Imbalanced Regression Problems

Several real world applications involve the prediction of a continuous target variable with different costs and benefits associated with the predictions. Frequently, in these problems, the higher costs and/or benefits are associated with the rarity of the target variable values. This is known as the problem of imbalanced domains. One of the challenges when dealing with imbalanced domains is the need of focusing the learners on rare cases. Several solutions have been proposed for addressing this challenge for classification tasks. However, this is a still scarcely explored issue in regression problems.

In this chapter we present several pre-processing solutions for tackling the problem of imbalanced domains in a regression context. An extensive experimental evaluation is carried out showing the advantages of the proposed approaches.

5.1 Introduction

The problem of imbalanced domains occurs in a diversity of predictive tasks, such as, regression, data streams, time series, multi-label, ordinal classification, multi-instance learning, among others [Branco et al., 2016b, Krawczyk, 2016]. Recently, the problem of learning from imbalanced domains in regression tasks has been gaining more attention. However, this is still a poorly studied problem with few solutions. In this chapter we focus our attention on imbalanced regression problems.

Learning from imbalanced domains poses important challenges, namely in what concerns the informal domain information, performance evaluation and learning process. The first two challenges were discussed in Chapter 3. In this chapter our goal is to propose solutions for addressing the problem of forcing the learners to focus on the most important, but rare,

cases. We developed pre-processing solutions that provide the flexibility of allowing the use of any standard learning algorithm to obtain the models after pre-processing takes place. Sections 5.2 and 5.3 present unbiased and biased pre-processing approaches for tackling imbalanced regression problems, respectively. This categorisation depends on the ability of the pre-processing method to consider or not the neighbourhood of the original examples. In Section 5.4 we present an extensive experimental evaluation of the proposals and Section 5.5 describes the main conclusions of the work described in this chapter.

5.2 Pre-processing Strategies for Imbalanced Regression Tasks

In this section, we propose solutions to solve the problem of performance degradation of the models on the most important cases when they are rare. The strategies that we propose are pre-processing or resampling methods that act before the learning procedure by changing the training data distribution. We will refer to these solutions as resampling or pre-processing methods.Details regarding the advantages and disadvantages of these solutions were presented in Table 2.14 (page 47). Pre-processing or resampling strategies can be clustered into three main types: i) under-sampling; ii) over-sampling; and iii) a mixture of both. Under-sampling involves removing uninteresting examples from the data set, while over-sampling increases the number of interesting examples.

The first challenge that we must face when tackling imbalanced regression problems is related with the definition of the interesting and uninteresting cases. This is an important issue because this will determine which cases to add or remove. This is a straightforward problem in a binary classification scenario: the most represented class or majority class will be under-sampled, while the least represented (minority) class will be over-sampled. In regression this decision is not so easy because the target variable is continuous. To address this problem we use the relevance function $\phi(Y)$ (defined in Section 3, page 62) and a threshold on the relevance values (t_R) to distinguish the training cases. The main idea is to build partitions with contiguous target variable values which belong either to $\mathcal{D}_R$ or $\mathcal{D}_N$ (defined in Section 3.4, page 89), i.e., each partition will include only examples with a high or low relevance based on the user provided threshold t_R. This procedure will allow us to obtain a set of bins with normal cases, denoted by $Bins_N$ and another set of bins with rare examples denoted by $Bins_R$. To build these data partitions we apply the following procedure:

i) sort the examples of the data set by ascending order of the target variable values;

ii) scan the data, starting from the lower Y value, and create a new bin whenever the relevance of the target variable value ($\phi(Y)$) changes from below to above the relevance threshold t_R or vice-versa.

This process generates a certain number of bins belonging either to $Bins_R$ or $Bins_N$. Each bin in $Bins_R$ contains only cases with $\phi(y) \geq t_R$ and contiguous target variable values. A similar reasoning applies to the bins in $Bins_N$. Algorithm 5.1 describes this method of generating bins.

Algorithm 5.1 Construction of Bins.

Input: $\mathcal{D}$ - original regression data set with target variable Y
t_R - threshold for relevance on Y values
$\phi(Y)$ - relevance function on Y values (Optional)

Output: $Bins$ - data set partitions into relevant and normal bins

1: **function** BINSCONSTRUCTOR($\mathcal{D}$, t_R, $\phi(Y)$)
2: $OrdD \leftarrow$ order $\mathcal{D}$ by ascending value of Y
3: **if** $\phi(Y) = \emptyset$ **then** ▷ Use the automatic method proposed by Ribeiro [2011].
4: $\phi() \leftarrow$ relevance function obtained from Y distribution
5: **end if**
6: $Bins_N \leftarrow k$ partitions of consecutive examples $\langle \mathbf{x}_i, y_i \rangle \in OrdD$, s.t. $\phi(y_i) < t_R$
7: $Bins_R \leftarrow l$ partitions of consecutive examples $\langle \mathbf{x}_i, y_i \rangle \in OrdD$, s.t. $\phi(y_i) \geq t_R$
8: $Bins \leftarrow Bins_R \bigcup Bins_N$
9: **return** $Bins$
10: **end function**

Figures 5.1 to 5.3 illustrate the impact in the bins obtained in $Bins_R$ and $Bins_N$ under different problem assumptions. We use the LNO2Emissions data set[1] and fixed the relevance threshold (t_R) at 0.8. In Figure 5.1 the used relevance function assumes that only the high extreme values of the target variable are relevant. Under this assumption, we observe that only one bin is generated for each of the sets $Bins_N$ and $Bins_R$.

However, if the used function assigns higher importance to both high and low extreme values of the target variable distribution then, with the same relevance threshold, we would obtain two distinct bins in $Bins_R$ and one bin in $Bins_N$ as displayed in Figure 5.2. The two bins with rare values are very different in this situation, and therefore, any over-sampling strategy applied to the rare cases should take this into account.

Finally, in Figure 5.3 we show another setting where we assume a relevance function that assigns high importance to the cases with target variable value below 0.8, between 2.35 and 3.2, and above 5.5. In this setting, Algorithm 5.1 returns 5 distinct bins: 3 bins in $Bins_R$ and 2 bins in $Bins_N$. We must highlight that the generation of bins is important for both the normal and rare cases. In either case, the bins obtained may have different characteristics, such as the number of cases or the range of values, that may be important when applying under- or over-sampling.

[1] Data set previously described in Section 3.3 in page 84.

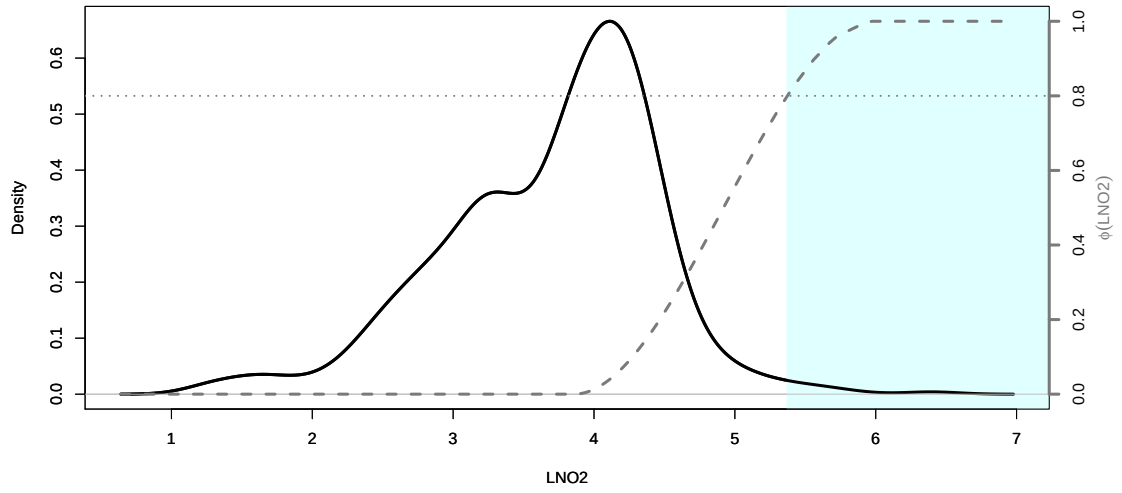


Figure 5.1: LNO2Emissions data with density target variable distribution (solid line) and corresponding relevance function $\phi()$ (dashed line). For $t_R = 0.8$ (horizontal dashed line), we have one bin in $Bins_R$ (blue shaded region) and another bin in $Bins_N$ (white region).

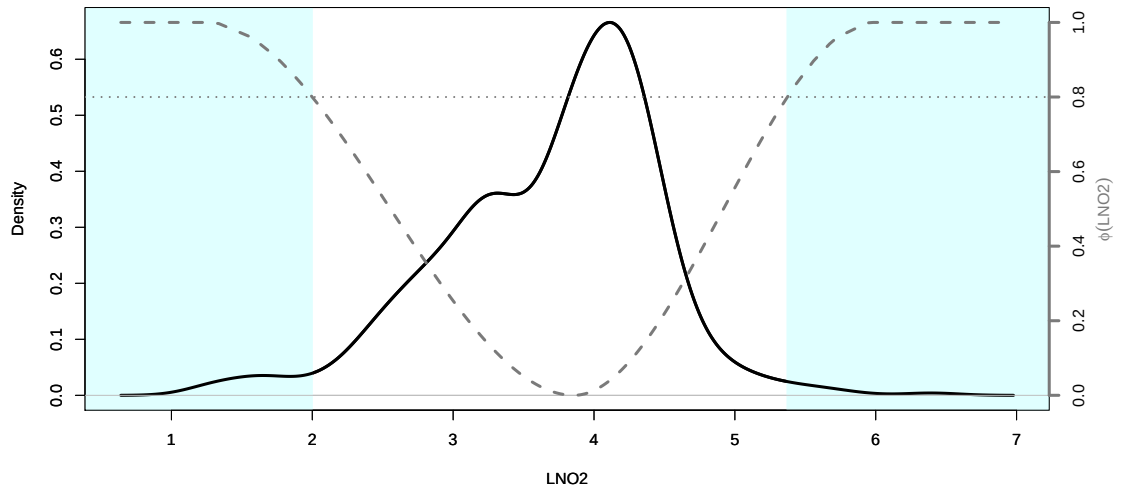


Figure 5.2: LNO2Emissions data with with density target variable distribution (solid line) and corresponding relevance function $\phi()$ (dashed line). For $t_R = 0.8$ (horizontal dashed line), we have two bins in $Bins_R$ (blue shaded region) and another bin in $Bins_N$ (white region).

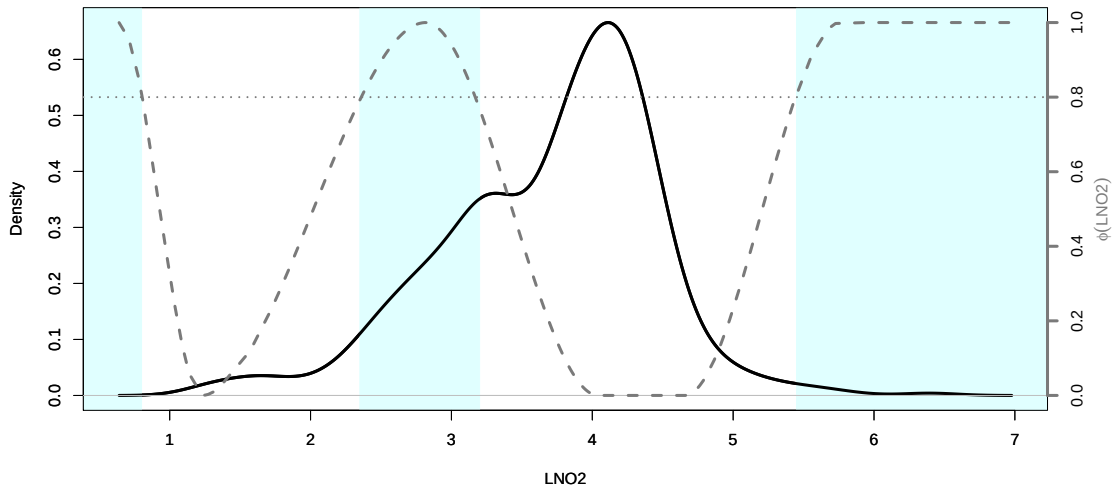


Figure 5.3: LNO2Emissions data with with density target variable distribution (solid line) and corresponding relevance function $\phi()$ (dashed line). For $t_R = 0.8$ (horizontal dashed line), we have three bins in $Bins_R$ (blue shaded region) and two bins in $Bins_N$ (white region).

Algorithm 5.1 solves the problem of deciding which are the candidate cases for over- and under-sampling while partitioning them by target variable value. This can be thought as a form of discretization of the continuous target variable. However, this method does not transform the initial regression task into a classification problem. It is simply used to decide which are the important and non-important cases according to the user preferences which are defined by a relevance function and a relevance threshold. Several other issues related with the continuous nature of the target variable remain to be decided, such as how to determine the target variable value when adding new examples.

The first pre-processing strategy that we propose is Random Under-sampling (RU), one of the simplest approaches for dealing with imbalanced domains. The key idea of under-sampling (e.g. Kubat and Matwin [1997]) is to remove examples with the most frequent target variable values for balancing the proportion of uninteresting and interesting examples in the available data set. Algorithm 5.2 describes the RU approach. This approach has two optional parameters: the relevance function and the percentage of under-sampling. When the relevance function is not provided by the user, the automatic method proposed by Ribeiro [2011] is applied. For the under-sampling percentage, a default setting of balancing the normal and rare cases is applied when this parameter is not set by the user. This procedure automatically determines the number of examples that each normal bin should have. This number is obtained by dividing the total number of rare cases by the number of normal bins. When this parameter is specified by the user, the number of removed cases is calculated

with respect to the number of existing normal cases in each normal bin. Parameter *u.perc* should be provided as a set of values defined in the interval $]0, 1[$, expressing the percentage of normal cases that are kept in each bin. If a single value is provided for parameter *u.perc* this percentage is used for all bins belonging to $Bins_N$. When the user specifies a set with different under-sampling percentages for each normal bin, they should be provided by increasing target variable value of those bins.

Algorithm 5.2 Random under-sampling (RU).

Input: $\mathcal{D}$ - regression data set with target variable Y
t_R - threshold for relevance on Y values
$\phi(Y)$ - relevance function on Y values (Optional)
u.perc - set of percentages of under-sampling (Optional)
Output: $NewD$ - a new modified data set

function RANDUNDER($\mathcal{D}$, t_R, $\phi(Y)$, *u.perc*)
  $Bins \leftarrow$ BINSCONSTRUCTOR($\mathcal{D}, t_R, \phi(Y)$)
  $Bins_R \leftarrow \{Bins_i \in Bins : \forall(\mathbf{x}, y) \in Bins_i, \phi(y) \geq t_R\}$
  $Bins_N \leftarrow Bins \setminus Bins_R$
  $NewD \leftarrow Bins_R$
  for $i \leftarrow 1$ **to** $|Bins_N|$ **do** ▷ Random under-sampling procedure
    if $|u.perc| = 0$ **then** ▷ Balance the normal and rare cases
      $TgtNr \leftarrow \frac{\sum_{\forall B \in Bins_R} |B|}{|Bins_N|}$
    else if $|u.perc| = 1$ **then** ▷ Same under-sampling percentage applied
      $TgtNr \leftarrow u.perc \times |Bins_N[i]|$
    else ▷ Each bin has a defined under-sampling percentage
      $TgtNr \leftarrow u.perc[i]$
    end if
    $selNormCases \leftarrow$ randomly sample $TgtNr$ elements from $Bins_N[i]$
    $newD \leftarrow newD \bigcup selNormCases$
  end for
return $NewD$
end function

Figure 5.4 shows the impact of applying the RU strategy with different parameter settings to the fuelCons data set[2].

The second pre-processing strategy that we propose for dealing with imbalanced regression tasks is Random Over-sampling (RO). This is also a simple and well-known strategy in the context of imbalanced classification tasks (e.g. Batista et al. [2004]). The key idea is to obtain a new data set where replicas of existing rare cases are added to the original data. The goal is again to better balance the distribution of rare and normal cases without

[2]Data set used in the Experimental Study Section were more details are presented.

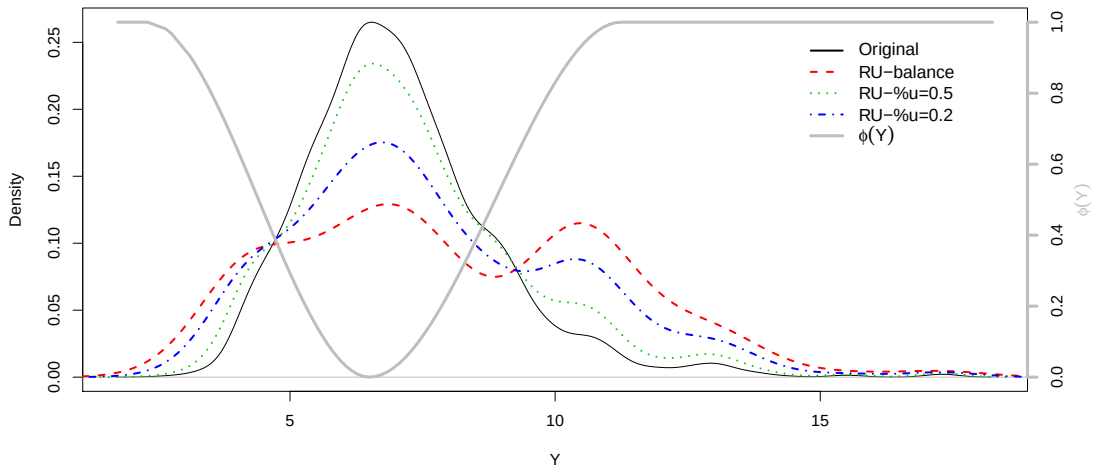


Figure 5.4: Density of the target variable on the original data set and after applying RU with different parameters on fuelCons data set ($\phi(Y)$ automatically estimated).

discarding any information. To achieve this, we use the BinsConstructor function defined in Algorithm 5.1 to obtain the sets of normal and rare bins, $Bins_N$ and $Bins_R$, respectively. In RO, the new data set contains all the original examples $\mathcal{D}$. Replicas of the examples in each bin of the set $Bins_R$, are also added to the new data set. The number of exact copies introduced in each bin in $Bins_R$ is determined by an optional parameter, *o.perc*. The default behaviour followed when parameter *o.perc* is not set by the user is to balance the number or rare and normal cases. The number of replicas to introduce in each bin is calculated dividing the total number of normal cases by the number of rare bins. If the user sets parameter *o.perc*, then the number of replicas to introduce is calculated with respect to the number of rare examples existing in each rare bin. Parameter *o.perc* should be greater than 1. When only one value for *o.perc* is provided this value is reused in all existing rare bins, otherwise the user should provide a value to apply to each rare bin. Algorithm 5.3 describes this pre-processing method. This method has the advantage of not discarding any of the original data. However, it leads to a larger data set and increases the likelihood of overfitting, specially when high over-sampling percentages are applied.

Figure 5.5 provides a brief overview of the impact of applying RO with different parameter settings in the density of the target variable of the fuelCons data set.

All the resampling approaches presented so far depend on the definition of important and non-important cases based on a relevance threshold, which can be thought as some sort of discretization of the target variable based on the relevance scores. Our next proposal, named **WE**ighted **R**elevance-based **C**ombination **S**trategy (WERCS), tries to avoid this step, by changing the data distribution solely based on the information of the relevance function.

Algorithm 5.3 Random over-sampling (RO).

Input: $\mathcal{D}$ - regression data set with target variable Y
t_R - threshold for relevance on Y values
$\phi(Y)$ - relevance function on Y values (Optional)
$o.perc$ - set of percentages of over-sampling (Optional)

Output: $NewD$ - a new modified data set

function RANDOVER($\mathcal{D}$, t_R, $\phi(Y)$, $o.perc$)
  $NewD \leftarrow \mathcal{D}$
  $Bins \leftarrow$ BINSCONSTRUCTOR$(\mathcal{D}, t_R, \phi(Y))$
  $Bins_R \leftarrow \{Bins_i \in Bins : \forall (\mathbf{x}, y) \in Bins_i, \phi(y) \geq t_R\}$
  $Bins_N \leftarrow Bins \setminus Bins_R$
  for $i \leftarrow 1$ **to** $|Bins_R|$ **do** ▷ Random over-sampling procedure
    if $|o.perc| = 0$ **then** ▷ Balance the normal and rare cases
      $TgtNr \leftarrow \frac{\sum_{\forall B \in Bins_N} |B|}{|Bins_R|}$
    else if $|o.perc| = 1$ **then** ▷ Same over-sampling percentage applied
      $TgtNr \leftarrow o.perc \times |Bins_R[i]|$
    else ▷ Each bin has a defined over-sampling percentage
      $TgtNr \leftarrow o.perc[i]$
    end if
    $selCases \leftarrow$ randomly sample $TgtNr$ elements from $Bins_R[i]$
    $NewD \leftarrow NewD \bigcup selCases$ ▷ Add the replicas to the new data set
  end for
return $NewD$
end function

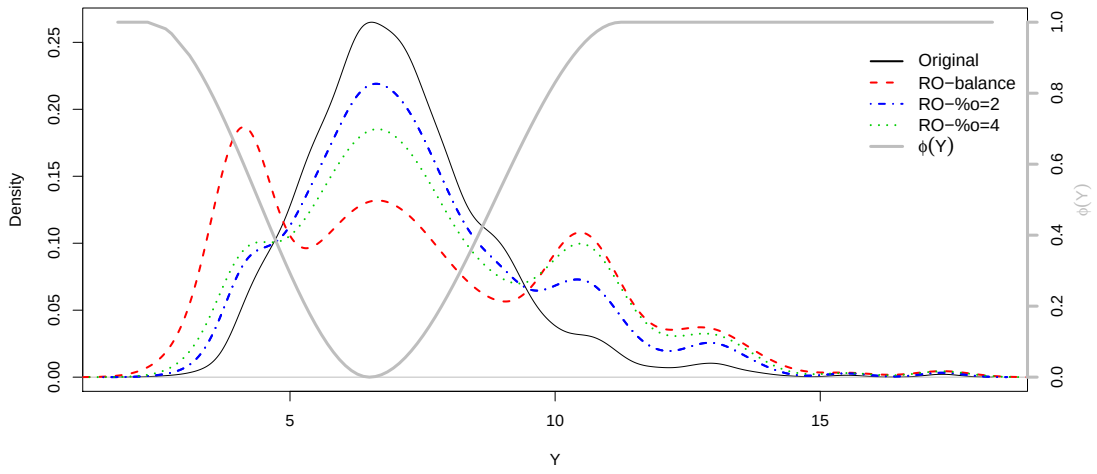


Figure 5.5: Density of the target variable on the original data set and after applying RO with different parameters on fuelCons data set ($\phi(Y)$ automatically estimated).

WERCS addresses the imbalanced regression problem through a combination of under- and over-sampling based on the relevance scores of the cases. Instead of randomly sampling either normal or rare cases based on a relevance threshold, WERCS biases this decision using the relevance values to derive weights for the examples, which are then used as a probability for selecting or not the examples. All examples are candidates for under-/over-sampling, the two steps that WERCS algorithm applies. Both strategies are applied in the entire domain. However, the probability of a case being selected for over- or under-sampling depends on the derived weights. This process is accomplished as follows:

- a percentage of cases is randomly selected to be added as replicas considering weights proportional to their respective relevance function values, i.e., $w(\langle \mathbf{x}_i, y_i \rangle) = \phi(y_i)$; and
- a percentage of cases is randomly selected for being removed considering weights that are the complement of their relevance function values, i.e., $w(\langle \mathbf{x}_i, y_i \rangle) = 1 - \phi(y_i)$.

Algorithm 5.4 implements this WERCS strategy. The way we derive the weights means that cases with higher relevance (and thus higher weight) have a higher probability of being replicated, whilst low relevance examples have higher probability of being removed. This facilitates the user task that only has to provide two percentages setting the level of under- and over-sampling to apply, parameters *u.perc* and *o.perc* respectively in Algorithm 5.4. Parameters *u.perc* and *o.perc* express the proportion of examples to replicate or to remove with respect to the number of examples in the original data set $\mathcal{D}$. Parameter $\phi(Y)$ is optional. When not set by the user, the method proposed by Ribeiro [2011] is used to obtain

$\phi(Y)$.

Algorithm 5.4 WEighted Relevance-based Combination Strategy (WERCS).

Input: $\mathcal{D}$ - regression data set with target variable Y
$\phi(Y)$ - relevance function on Y values (Optional)
$u.perc$ - percentage of under-sampling
$o.perc$ - percentage of over-sampling

Output: $newD$ - a new modified data set

1: **function** WERCS($\mathcal{D}$, $\phi(Y)$, $u.perc$, $o.perc$)
2: **if** $\phi(Y) = \emptyset$ **then** ▷ Use the automatic method proposed by Ribeiro [2011].
3: $\phi() \leftarrow$ relevance function obtained from Y distribution
4: **end if**
5: $newD \leftarrow \mathcal{D}$
6: $WOve \leftarrow \{\phi(y_i) \mid y_i \in Y\}$
7: $Ove \leftarrow$ sample $o.perc \times |\mathcal{D}|$ cases from $\mathcal{D}$ with $WOve$ weights ▷ over-sampling procedure
8: $newD \leftarrow newD \bigcup Ove$
9: $WUnd \leftarrow \{1 - \phi(y_i) \mid y_i \in Y\}$
10: $Und \leftarrow$ sample $u.perc \times |\mathcal{D}|$ cases from $\mathcal{D}$ with $WUnd$ weights
11: $newD \leftarrow newD \setminus Und$ ▷ under-sampling procedure
return $newD$
12: **end function**

The most interesting feature of the WERCS approach is that takes into account the intrinsic continuous nature of the target variable, avoiding a crisp partition of the data into bins. Let us consider, for instance, two examples A and B with relevance value of 0.8 and 0.9, respectively. In WERCS these examples would have a similar probability of being over- or under-sampled because they have similar relevance values, and thus will have similar weights. However, in the context of a strategy using bins, if for instance the selected relevance threshold was 0.85, these two examples would be assigned to distinct bins: A to $Bins_N$ and B to $Bins_R$. This means that example A would be candidate for an under-sampling strategy with the same probability as examples with relevance zero, while B would be a candidate for over-sampling with the same probability as all other cases with high relevance. WERCS prevents this counter-intuitive procedure by avoiding the use of the relevance threshold, and by using the relevance scores to perform informed under-/over-sampling.

Figure 5.6 displays the impact of choosing different parameter settings for the WERCS algorithm on the density distribution of the target variable of the fuelCons data set.

The next pre-processing strategies we will describe are different from the previous ones because they all generate new synthetic examples. The main motivation for generating new synthetic cases is to face the overfitting likelihood that may occur when simple replicas of

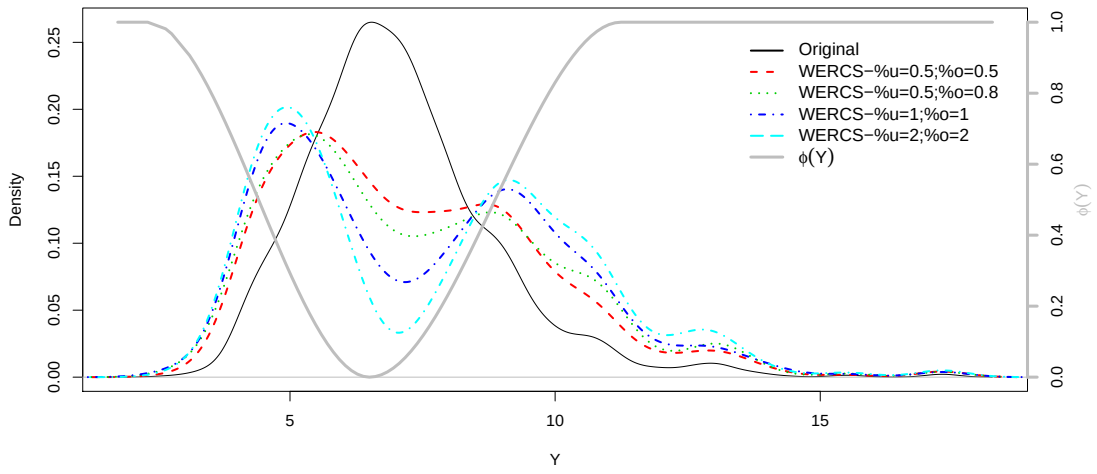


Figure 5.6: Density of the target variable on the original data set and after applying WERCS with different parameters on fuelCons data set ($\phi(Y)$ automatically estimated).

existing examples are used as in over-sampling procedures.

The first of these strategies we propose is the Introduction of Gaussian Noise (GN). This strategy changes the data distribution through two main steps:

- under-sampling the normal and less important cases in $Bins_N$; and
- generating new cases with relevant target variable values by adding Gaussian Noise to the feature values of original rare cases.

The process for generating new cases uses an adaptation of the method proposed in Lee [1999, 2000] for classification tasks. Lee [2000] described an over-sampling strategy that obtains new minority class cases using the addition of normally distributed noise to existing minority class cases. We use a similar strategy for over-sampling some cases in the bins in $Bins_R$. Namely, given a seed rare case, we obtain new cases from it by introducing small perturbations on both the attributes and the target variable value of the seed case.

Algorithm 5.5 describes the GN method. The algorithm starts by applying random under-sampling to the normal cases as defined in Algorithm 5.2. With respect to the over-sampling strategy, GN first determines the number of examples to be generated in each bin belonging to $Bins_R$. This is calculated using the optional parameter *o.perc*. If this parameter is not provided the algorithm will balance the number of rare and normal cases. If it is provided, then the specified over-sampling percentage is applied.

The new synthetic cases are generated as follows:

- for the numeric attributes and the target variable, we add a perturbation randomly drawn from a normal distribution $N(0, \delta \times sd(a))$, where δ is a user-defined parameter controlling the amplitude of the perturbation, and $sd(a)$ is the standard deviation of attribute a estimated using the cases in the bin under consideration;

- for nominal attributes, sample from the values in the bin with a probability proportional to the frequency of the values in the bin.

In the generation of the target variable value the algorithm makes sure the new synthetic cases are generated inside the high relevance bin.

Figure 5.7 provides an overview of the impact on the density distribution of the target variable of the fuelCons data set for different parameter settings of the GN algorithm.

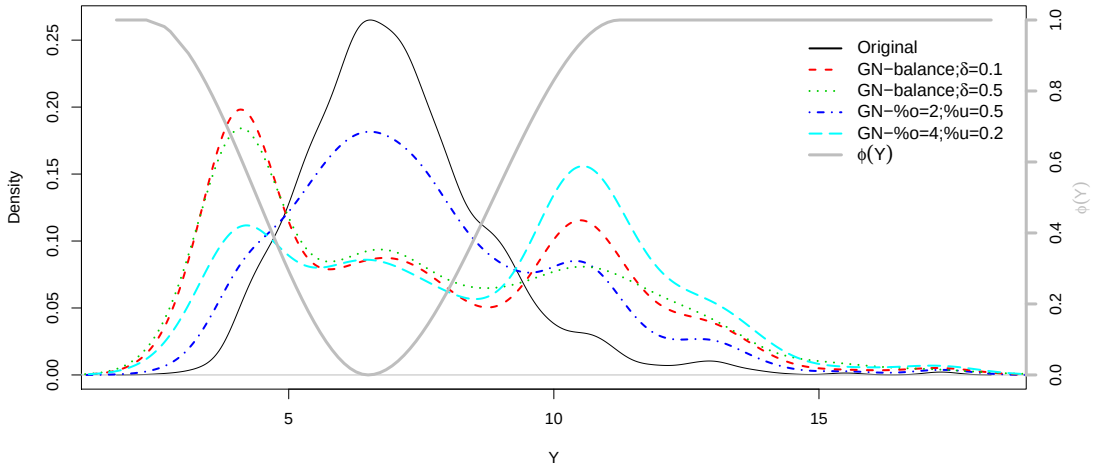


Figure 5.7: Density of the target variable on the original data set and after applying GN with different parameters on fuelCons data set ($\phi(Y)$ automatically estimated).

The next proposed strategy, named SMOTE for Regression (SMOTER), also generates synthetic examples and is based on the well-known SMOTE algorithm [Chawla et al., 2002]. The SMOTER algorithm was proposed by Torgo et al. [2013, 2015] for dealing with imbalanced regression problems with one normal and two relevant bins at most.

We extended the SMOTER algorithm so that it becomes able to deal with any number of normal and rare bins. The SMOTER algorithm begins by using the function BinsConstructor defined in Algorithm 5.1 to obtain the sets of bins $Bins_N$ and $Bins_R$. Then, if the user did not specify parameters *u.perc* and *o.perc*, the algorithm defaults to the strategy of balancing all bins while maintaining the total number of examples in the data set. The next step involves applying random under-sampling (cf. Algorithm 5.2). Finally, the over-

Algorithm 5.5 Introduction of Gaussian Noise (GN).

Input: $\mathcal{D}$ - regression data set with target variable Y
t_R - threshold for relevance on Y values
$\phi(Y)$ - relevance function on Y values (Optional)
u.perc, *o.perc* - sets with percentages of under- and over-sampling (Optional)
δ - perturbation amplitude

Output: *newD* - a new modified data set

function GaussNoiseIntroduction($\mathcal{D}$, t_R, $\phi(Y)$, *u.perc*, *o.perc*, δ,)
  $Bins \leftarrow$ BinsConstructor($\mathcal{D}, t_R, \phi(Y)$)
  $Bins_N \leftarrow \{Bins_i \in Bins : \forall(\mathbf{x}, y) \in Bins_i, \phi(y) < t_R\}$
  $Bins_R \leftarrow Bins \setminus Bins_N$
  if $|u.perc| = |o.perc| = 0$ **then**
    update *u.perc* and *o.perc* in order to balance all the bins and maintain the data set size
  end if
  $newD \leftarrow$ RandUnder($\mathcal{D}, t_R, \phi(Y), u.perc$) ▷ under-sampling procedure
  for $j \leftarrow 1$ **to** $|Bins_R|$ **do** ▷ over-sampling procedure
    if $|o.perc| = 1$ **then** ▷ Determine the number of examples to generate in the bin
      $TgtNr \leftarrow o.perc \times |Bins_R[j]|$
    else
      $TgtNr \leftarrow o.perc[j]$
    end if
    for each $case \in Bins_R[j]$ **do** ▷ generate synthetic examples
      for $i \leftarrow 1$ **to** $TgtNr$ **do**
        for each $a \in Attrs \bigcup Y$ **do**
          if a is nominal **then**
            $probs \leftarrow$ frequency of possible values of a
            $new[a] \leftarrow$ sample a value from the values of a with weights = *probs*
          else
            $sp \leftarrow$ random sample from $N(0, \delta \times sd(a))$
            $new[a] \leftarrow case[a] \times sp$
          end if
        end for
        $newD \leftarrow newD \bigcup \{new\}$ ▷ add synthetic case to *newD*
      end for
    end for
  end for
return *newD*
end function

sampling procedure is applied on the bins belonging to $Bins_R$. Algorithm 5.7 describes the SMOTER strategy and Algorithm 5.6 shows the procedure for generating synthetic cases.

Algorithm 5.6 Generating synthetic cases in regression (GenSynthCases).

Input: $\mathcal{D}$ - set of base cases from which new synthetic cases are generated
o - percentage of over-sampling
k - number of neighbours used in case generation

Output: $newCases$ - a set of new synthetic cases

function GENSYNTHCASES($\mathcal{D}$, o, k)
  $newCases \leftarrow \{\}$
  $ng \leftarrow (o-1) \times |\mathcal{D}|$ ▷ nr. of new cases to generate for each existing case
  for each $case \in \mathcal{D}$ **do**
    $nns \leftarrow \text{KNN}(k, case, \mathcal{D} \setminus \{case\})$ ▷ k-Nearest Neighbours of $case$
    for $i = 1$ **to** ng **do**
      $x \leftarrow$ randomly choose one of the nns
      for each $a \in$ attributes **do** ▷ Generation of the attribute values
        if IS.NUMERIC(a) **then**
          $diff \leftarrow case[a] - x[a]$
          $new[a] \leftarrow case[a] + \text{RANDOM}(0,1) \times diff$
        else
          $new[a] \leftarrow$ randomly select among $case[a]$ and $x[a]$
        end if
      end for ▷ Generation of the target value
      $d_1 \leftarrow \text{DIST}(new, case)$
      $d_2 \leftarrow \text{DIST}(new, x)$
      $new[Target] \leftarrow \frac{d_2 \times case[Target] + d_1 \times x[Target]}{d_1 + d_2}$
    end for
    $newCases \leftarrow newCases \bigcup new$ ▷ Add the new synthetic case
  end for
return $newCases$
end function

The over-sampling procedure of SMOTER algorithm is an adaptation to regression of the mechanism proposed in SMOTE for classification. This procedure generates the features of a new synthetic case by interpolating a seed rare example with one of its k Nearest Neighbours (k-NN). The target variable value of the synthetic case is calculated as a weighted average between the target variable values of the two original cases being interpolated. The higher is the distance to the case, the lower will be the weight.

Figure 5.8 shows the impact on the target variable density distribution of the fuelCons data when different parameter values are selected for applying the SMOTER algorithm.

Algorithm 5.7 SMOTE for Regression (SMOTER).

Input: $\mathcal{D}$ - regression data set with target variable Y
t_R - threshold for relevance on Y values
$\phi(Y)$ - relevance function on Y values (Optional)
u.perc, *o.perc* - sets with percentages of under- and over-sampling (Optional)
k - number of used neighbours in case generation

Output: *newD* - a new modified data set

function SMOTER($\mathcal{D}$, t_R, $\phi(Y)$, *u.perc*, *o.perc*, k)
  $Bins \leftarrow$ BinsConstructor$(\mathcal{D}, t_R, \phi(Y))$
  $Bins_N \leftarrow \{Bins_i \in Bins : \forall(\mathbf{x}, y) \in Bins_i, \phi(y) < t_R\}$
  $Bins_R \leftarrow Bins \setminus Bins_N$
  if $|u.perc| = |o.perc| = 0$ **then**
    update *u.perc* and *o.perc* in order to balance all the bins and maintain the data set size
  end if
  $newD \leftarrow$ RandUnder$(\mathcal{D}, t_R, \phi(Y), u.perc)$ ▷ under-sampling procedure
  for $i \leftarrow 1$ **to** $|Bins_R|$ **do** ▷ over-sampling procedure
    if $|o.perc| = 1$ **then**
      $newCases \leftarrow$ genSynthCases$(Bins_R[i], o.perc, k)$
    else
      $newCases \leftarrow$ genSynthCases$(Bins_R[i], o.perc[i], k)$
    end if
    $newD \leftarrow newD \bigcup newCases$
  end for
return *newD*
end function

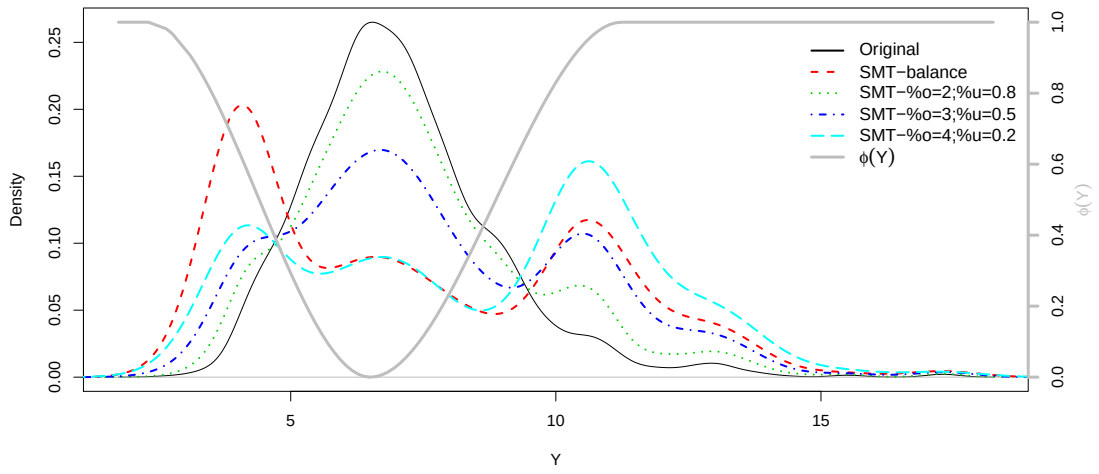


Figure 5.8: Density of the target variable on the original data set and after applying SMOTER with different parameters on fuelCons data set ($\phi(Y)$ automatically estimated).

The last strategy proposed in this section is the SMOTER with Gaussian Noise (SMOGN) algorithm. This method combines the application of random under-sampling to the normal cases with two distinct over-sampling strategies: SMOTER and GN. The key idea of SMOGN is to combine both strategies for generating synthetic examples with the goal of simultaneously: i) limiting the risks that SMOTER can incur into by using the more conservative strategy of GN, and ii) allow an increase of the diversity in examples generation, which is not possible to achieve when using only the GN. SMOGN generates synthetic examples with SMOTER only when the seed example and the k-NN selected are "close enough" and will apply GN strategy to the seed example when this example and its nearest neighbour are too far apart, which would increase the risk of the interpolation. Figure 5.9 shows an example of the application of SMOGN. The relevant cases are represented by blue dots and the normal cases are represented by green crosses. In this figure, for a given case we marked its 5-NNs. For a certain threshold on the distances, we can see that only three of those neighbours are considered "safe". Therefore, SMOGN algorithm will generate new synthetic cases with SMOTER for any of these neighbours. The two more distant neighbours are considered "unsafe", and will not be used in the interpolation process. Instead, SMOGN will follow a more conservative procedure by applying GN on the seed example.

Algorithm 5.8 describes the SMOGN Algorithm. This algorithm also includes two optional parameters, *u.perc* and *o.perc*, representing the sets with the percentages of over and under sampling to apply. Similar to the previously presented algorithms, when these parameters are not set by the user, the distribution will be balanced across all bins in $Bins_R$ and $Bins_N$ maintaining at the same time the data set size. Parameter $\phi(Y)$ is also optional,

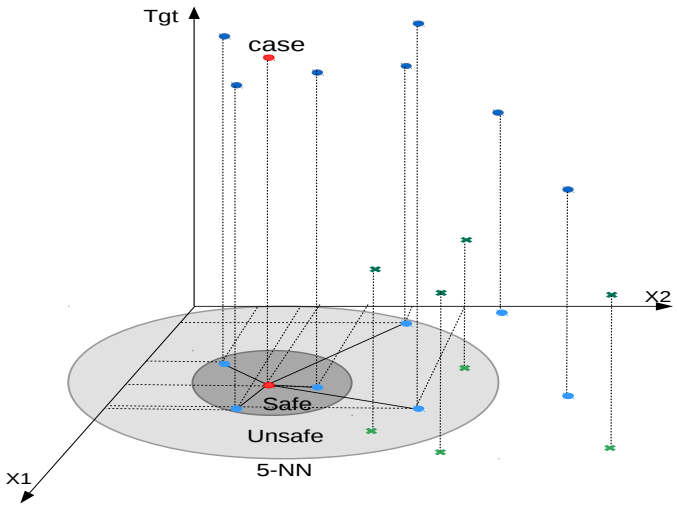


Figure 5.9: Synthetic example illustrating the application of SMOGN algorithm (blue dots represent rare cases and green crosses represent normal cases).

being estimated through a distribution derived method as proposed by Ribeiro [2011], if not supplied by the user.

After building the sets $Bins_R$ and $Bins_N$, SMOGN applies random under-sampling to the normal cases in $Bins_N$, while the bins in $Bins_R$ will be subject to an over-sampling procedure. For each case (the seed example) in a bin belonging to $Bins_R$, a number of synthetic cases is generated. The over-sampling will use either the SMOTER or GN strategy to generate new cases depending on the distance between the seed example and the selected k-NN. The main idea is that, if the selected neighbour is "safe", then he is in a distance considered to be suitable to perform interpolation through SMOTER. On the other hand, if the selected neighbour is not in a range considered safe, then he is too far away to be used to perform interpolation which means that, in this case, it is better to generate a new example by applying GN on the seed case. The threshold that is used to decide if the neighbour is at a safe or unsafe distance depends on the distance between the seed example and all the remaining cases in the partition under consideration. We used half of the median of the distances between the seed example and the other examples in the same partition.

Figure 5.10 displays the impact on the density distribution of fuelCons data set target variable for different parameter settings of SMOGN algorithm.

Algorithm 5.8 SMOTER with Gaussian Noise (SMOGN).

Input: $\mathcal{D}$ - regression data set with target variable Y
t_R - threshold for relevance on Y values
$\phi(Y)$ - relevance function on Y values (Optional)
u.perc, *o.perc* - sets with percentages of under- and over-sampling (Optional)
k - number of used neighbours in case generation
δ - perturbation amplitude

Output: *newD* - a new modified data set

function SMOGN($\mathcal{D}$, t_R, $\phi(Y)$,*u.perc*, *o.perc*, k, δ)
  $Bins \leftarrow$ BinsConstructor($\mathcal{D}, t_R, \phi(Y)$)
  $Bins_N \leftarrow \{Bins_i \in Bins : \forall(\mathbf{x}, y) \in Bins_i, \phi(y) < t_R\}$
  $Bins_R \leftarrow Bins \setminus Bins_N$
  if $|u.perc| = |o.perc| = 0$ **then**
    update *u.perc* and *o.perc* in order to balance all the bins and maintain the data set size
  end if
  $newD \leftarrow$ RandUnder($\mathcal{D}, t_R, \phi(Y), u.perc$) ▷ under-sampling procedure
  for $j \leftarrow 1$ **to** $|Bins_R|$ **do** ▷ over-sampling procedure
    if $|o.perc| = 1$ **then** ▷ Determine the number of examples to generate in the bin
      $TgtNr \leftarrow o.perc \times |Bins_R[j]|$
    else
      $TgtNr \leftarrow o.perc[j]$
    end if
    for each $case \in Bins_R[j]$ **do** ▷ generate synthetic examples
      $nns \leftarrow kNN(k, case, Bins_R[j])$ ▷ k-Nearest Neighbours of case
      $DistM \leftarrow$ distances between the case and the examples in $Bins_R[j]$
      $maxD \leftarrow median(DistM)/2$
      for $i \leftarrow 1$ **to** $TgtNr$ **do**
        $x \leftarrow$ randomly choose one of the nns
        if $DistM(x) < maxD$ **then** ▷ safe kNN selected
          $new \leftarrow$ use SMOTER to interpolate x and *case*
        else ▷ non-safe kNN selected
          $pert \leftarrow min(maxD, \delta)$
          $new \leftarrow$ introduce Gaussian Noise in *case* with a perturbation *pert*
        end if
        $newD \leftarrow newD \bigcup \{new\}$ ▷ add synthetic case to newD
      end for
    end for
  end for
return *newD*
end function

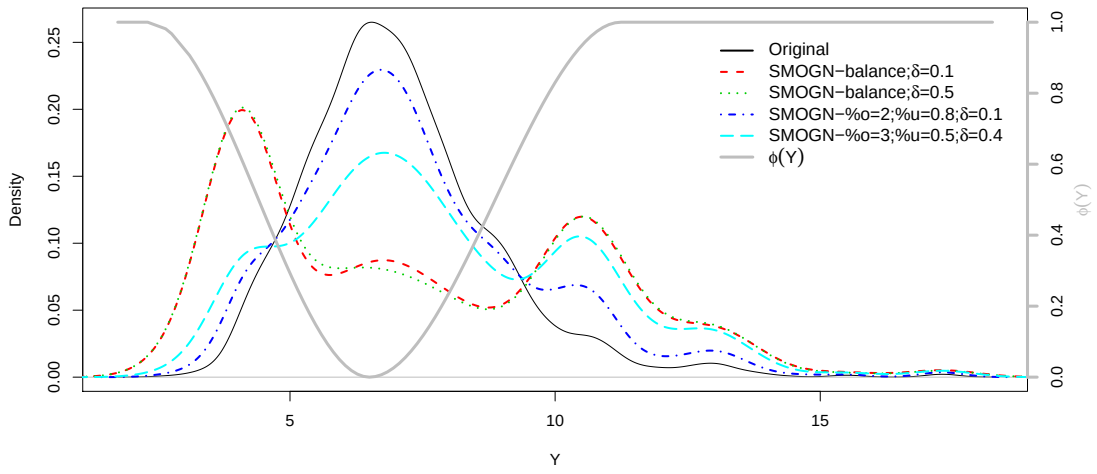


Figure 5.10: Density of the target variable on the original data set and after applying SMOGN with different parameters on fuelCons data set ($\phi(Y)$ automatically estimated).

5.3 Biased Pre-processing Strategies for Imbalanced Regression Tasks

In the previous section we presented resampling strategies for dealing with problem of imbalanced regression. These strategies include the removal of normal and uninteresting examples and/or the inclusion of rare cases by replication or synthetic generation.

Nevertheless, the procedure for removing or adding cases is unbiased w.r.t. the neighbourhood of the examples. These strategies either randomly remove or add cases, or follow a method that does not take into consideration whether the neighbours of the seed examples (for removal or generation of new cases) are interesting or uninteresting. This may be crucial because the analysis of a case neighbourhood may reveal important characteristics. For instance, a rare case can be completely surrounded by normal cases, or the opposite may happen. This analysis may serve as guidance for the under-/over-sampling strategy which may use this knowledge to apply a more informed procedure. In effect, the characteristics of the neighbourhood may provide important indications, like for instance if it is too risky to interpolate between two interesting cases because they are surrounded by uninteresting cases. This and other properties of a case neighbourhood are important to be checked to make sure the over- and/or under-sampling procedures do help the learning process instead of causing unwanted performance degradation. The key distinguishing feature of the biased strategies presented in this section is to actively use the neighbourhood knowledge available to bias the resampling procedures for reinforcing specific regions of the domain.

We present two algorithms for biasing an under-sampling and an over-sampling strategy by using the information of each case neighbourhood. Our goal is to act on specific domain regions that may be more responsible for a degradation in the learners' performance. Instead of treating all rare (or normal) cases uniformly, by inspecting the examples neighbourhood, we will be able to apply a differentiated resampling method across the problem domain.

Let us begin with two definitions that may be applied to both normal and rare cases.

Definition 5.3.1 (Degree of Closeness to Frontier) *Let us consider an example* $ex_i = \langle \mathbf{x}_i, y_i \rangle \in \mathcal{D}$ *that belongs to a certain bin* B *contained in either* $Bins_R$ *or* $Bins_N$*. Determine the* k *nearest neighbours of example* $ex_i \in \mathcal{D}$*. The degree of closeness to frontier of case* ex_i *is the proportion of these k-nearest neighbours that do not belong to the same bin* B*,*

$$DF(ex_i) = \frac{|kNN(ex_i) \notin B|}{k} \tag{5.1}$$

where $kNN(ex_i)$ *is the set of* k *nearest neighbours of the case* ex_i*.*

Definition 5.3.2 (Degree of Safeness) *Consider an example* $ex_i = \langle \mathbf{x}_i, y_i \rangle \in \mathcal{D}$ *that belongs to a certain bin* B *contained in either* $Bins_R$ *or* $Bins_N$*. The degree of safeness of case* ex_i *is the proportion of its k-nearest neighbours that belong to the same bin* B*,*

$$DS(ex_i) = \frac{|kNN(ex_i) \in B|}{k} = 1 - DF(ex_i) \tag{5.2}$$

The two presented properties are complementary, the higher is the degree of closeness to the frontier the lower is the safeness degree, and vice-versa.

Cases with high degrees of closeness to the frontier have as their nearest neighbours cases that belong to a different bin. These cases can be regarded as hard to learn or outliers, as they are surrounded by cases of different type. On the other hand, cases with a low degree of closeness to the frontier, and thus a high degree of safeness, have most of their k-nearest neighbours in the same bin, and therefore, can be seen as easier to learn.

Figure 5.11 shows the two extreme situations of cases with a maximum degree of closeness to the frontier and safeness applied to both rare and normal cases when considering 3-nearest neighbours. In this figure, the higher target variable values represent the rare and important cases, i.e., the set $\mathcal{D}_R$, while the low target variable values represent the set $\mathcal{D}_N$. For simplicity purposes, let us assume that the partition into bins produced only two bins that match the sets $\mathcal{D}_R$ and $\mathcal{D}_N$. We used the colours red and orange for representing the cases with higher degree of closeness to the frontier and the cases with higher degree of safeness, respectively. We highlight that cases with different degrees of safeness and closeness to frontier may arise when the cases neighbourhood has a mixture of normal and rare cases.

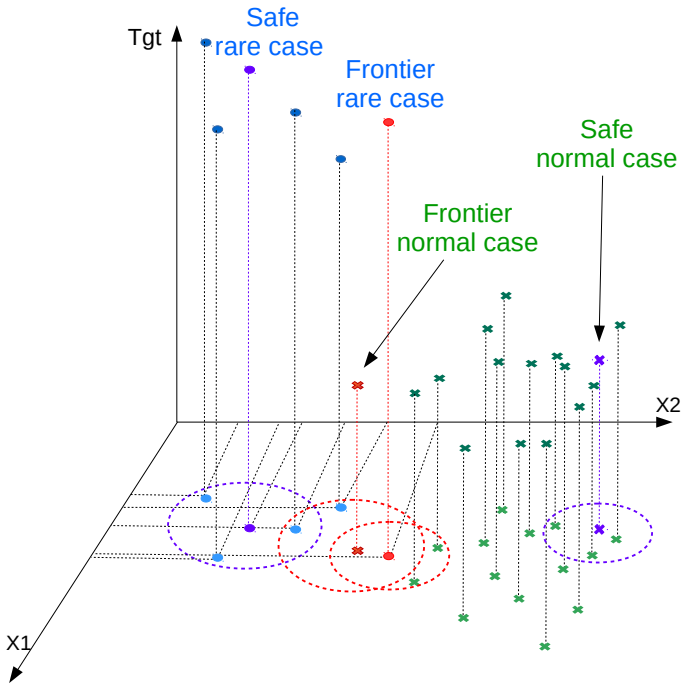


Figure 5.11: Illustration of cases with the higher degree of closeness to frontier (red) and higher degree of safeness (purple) for both rare (blue) and normal (green) examples in a regression problem. The notions of degree of closeness to frontier and degree of safeness were assessed based on 3-nearest neighbours.

The two following variants may be applied for biasing pre-processing strategies according to the cases neighbourhood: i) **reinforce the frontier**, or harder to learn cases; and/or ii) **reinforce the safe** or easier to learn cases. The first variant reinforces with higher probability the cases with a higher degree of closeness to the frontier. On the other hand, the second strategy reinforces the cases with higher degree of safeness, i.e., the higher the safeness degree of a case the more likely it will be selected to be reinforced. The degree of safeness or closeness to the frontier of a case, being a number in $[0, 1]$ interval, is used as a sampling probability of the case. This sampling probability can then be used to apply either under- or over-sampling to certain regions. These regions are determined by the selected bias variant that may reinforce the frontier or the safe regions.

Reinforcement of the frontier regions can be accomplished through over or under-sampling. When performing over-sampling synthetic examples are generated with higher probability for cases with a high degree of closeness to the frontier. On the other hand, when applying under-sampling to reinforce the frontier, examples with higher degree of closeness to the frontier are more likely to be kept. In both situations, the bias will favour the cases that are closer to the frontier.

The same reasoning applies to the strategy that reinforces the safer cases. In this case it is also possible to apply both under- and over-sampling. However, the probability for generating new cases or removing cases is now related with the degree of safeness of the cases.

Algorithms 5.9 and 5.10 describe the proposed variants for biasing the resampling strategies. These algorithms require as input a bin B obtained using Algorithm 5.1. The bin should belong to $Bins_N$ when an under-sampling strategy is applied, and to $Bins_R$ when over-sampling is applied. This happens because we only apply under-sampling strategies to the normal cases and over-sampling to the rare cases. Both algorithms return a new changed bin. Both algorithms calculate, for each case $\mathbf{x}_i$, a value r_i that expresses either the degree of closeness to the frontier or the safeness of the case, depending on the biasing strategy applied.

Algorithm 5.9 requires the setting of the number of examples that the modified bin should have. It also requires the definition of a logical parameter Fr that sets if reinforcement is applied to the closest to the frontier or safer cases.

Algorithm 5.10 requires that the user defines how many examples should be generated in the new modified bin. It also has a logical parameter, Fr, that determines whether the reinforcement should be applied in the frontier or in the safe cases. This algorithm requires the definition of parameter $GenEx$, which sets the method that should be applied for obtaining new examples. This parameter can be set to any procedure that generates new cases such as: introduction of replicas, use of GN or interpolation with SMOTER.

Algorithm 5.9 Under-sampling with neighbourhood bias (U_Bias).

Input: $\mathcal{D}$ - regression data set with target variable Y

B - a bin belonging to $Bins_N$ with normal cases provided by algorithm 5.1

$tgtNr$ - target number of examples to obtain in the new bin

k - number of evaluated neighbours

Fr - logical value indicating if the reinforcement is applied to the frontier (TRUE) or to the safe (FALSE) cases

Output: $newB$ - a new modified data bin

function UndNeigBias($\mathcal{D}$, B, $tgtNr$, k, Fr)
  $KNNs \leftarrow kNN(B, D, k)$ ▷ k-NN of all cases in B evaluated in set $\mathcal{D}$
  $\boldsymbol{r} \leftarrow$ vector of dimension $|B|$
  for each $ex_i \in B$ **do**
    if Fr = TRUE **then**
      $\Delta_i \leftarrow$ nr of KNNs of ex_i that belong to B
    else
      $\Delta_i \leftarrow$ nr of KNNs of ex_i that do not belong to B
    end if
    $r_i \leftarrow \Delta_i / k$
  end for
  $\hat{\boldsymbol{r}} \leftarrow \boldsymbol{r} / \sum_{i=1}^{|B|} r_i$
  $newD \leftarrow$ sample $tgtNr$ examples from B with sampling probability $\hat{\boldsymbol{r}}$
  return $newD$
end function

Algorithm 5.10 Over-sampling with neighbourhood bias (O_Bias).

Input: $\mathcal{D}$ - regression data set with target variable Y

B - a bin belonging to $Bins_R$ with relevant cases provided by algorithm 5.1

$tgtNr$ - number of new examples to generate in the new bin

k - number of evaluated neighbours

Fr - logical value indicating if the reinforcement is applied to the frontier (TRUE) or safe (FALSE) cases

$GenEx$ - function for obtaining the new examples

Output: $newB$ - a new modified data bin

1: **function** OverNeigBias($\mathcal{D}$, B, $tgtNr$, k, Fr, $GenEx$)
2: $KNNs \leftarrow kNN(B, D, k)$ ▷ k-NN in set $\mathcal{D}$ of examples in B
3: $\boldsymbol{r} \leftarrow$ vector of dimension $|B|$
4: **for each** $ex_i \in B$ **do**
5: **if** Fr = TRUE **then**
6: $\Delta_i \leftarrow$ nr of KNNs of ex_i that do not belong to B
7: **else**
8: $\Delta_i \leftarrow$ nr of KNNs of ex_i that belong to B
9: **end if**
10: $r_i \leftarrow \Delta_i / k$
11: **end for**
12: $\hat{\boldsymbol{r}} \leftarrow \boldsymbol{r} / \sum_{i=1}^{|B|} r_i$
13: **for** $i = 1$ **to** $|B|$ **do**
14: $g_i \leftarrow \hat{r}_i \times tgtNr$
15: **end for**
16: $newD \leftarrow$ use $GenEx$ function to generate g_i new examples for each ex_i
17: **return** $newD$
18: **end function**

From an end-user perspective, the decision of either reinforcing the frontier or the safe cases may not be trivial. The better option can be data-dependent and several arguments may be put forward for and against the two options. For instance, in a noisy data set it may be probably better to generate new rare examples based on the existing safe rare cases. However, if we have a data set with few noisy examples, then, the use of the frontier cases for obtaining new cases may be beneficial.

5.4 Experimental Study

In this section we provide an experimental analysis of the pre-processing strategies we have previously presented. We study several different aspects related with the impact of applying the biased and unbiased pre-processing strategies in the learning algorithms performance. The main research questions that we aim to answer are:

- How do the unbiased pre-processing strategies impact the performance of the learning algorithm? (Section 5.4.2)
- What is the impact of applying biasing pre-processing strategies in comparison to the use of unbiased strategies? (Section 5.4.3);
- What is the impact of applying a change in the training data distribution that does not aim at balancing the examples distribution? (Section 5.4.4).

Section 5.4.1 describes the experimental setup used on the experiments carried out for answering to the two first questions. In Section 5.4.4 we describe the setup used in answering the third question that is different with respect to the used learning algorithms.

5.4.1 Materials and Methods

Data Sets

We have used 20 real world regression data sets in our experimental study. For each of these data sets we estimated the relevance function, $\phi(Y)$, using the automatic method proposed by Ribeiro [2011]. We set the relevance threshold to 0.8 to define the sets $\mathcal{D}_R$ and $\mathcal{D}_N$ of rare and normal cases. According to this setting, the used data sets have a percentage of rare cases that ranges between 20.4% and 4.1%. The main characteristics of the used data sets are shown in Table 5.1.

Table 5.1: Data sets information by descending percentage of rare cases. (N: nr. of cases; *tpred*: nr. of predictors; *p.nom*: nr. of nominal predictors; *p.num*: nr. of numeric predictors; $nRare$: nr. of cases with $\phi(y) > 0.8$; $\%Rare$: $nRare/N \times 100$).

ID	**Data Set**	**N**	**tpred**	**p.nom**	**p.num**	**nRare**	**%Rare**
DS1	servo	167	4	2	2	34	20.4
DS2	a6	198	11	3	8	33	16.7
DS3	Abalone	4177	8	1	7	679	16.3
DS4	machCpu	209	6	0	6	34	16.3
DS5	a3	198	11	3	8	32	16.2
DS6	a4	198	11	3	8	31	15.7
DS7	a1	198	11	3	8	28	14.1
DS8	a7	198	11	3	8	27	13.6
DS9	boston	506	13	0	13	65	12.8
DS10	a2	198	11	3	8	22	11.1
DS11	a5	198	11	3	8	21	10.6
DS12	fuelCons	1764	37	12	25	164	9.3
DS13	availPwr	1802	15	7	8	157	8.7
DS14	cpuSm	8192	12	0	12	713	8.7
DS15	maxTorq	1802	32	13	19	129	7.2
DS16	bank8FM	4499	8	0	8	288	6.4
DS17	dAiler	7129	5	0	5	450	6.3
DS18	ConcrStr	1030	8	0	8	55	5.3
DS19	Accel	1732	14	3	11	89	5.1
DS20	airfoild	1503	5	0	5	62	4.1

Learning Algorithms

In order to avoid the introduction of any bias in our conclusions due to the used modelling techniques, we selected four different types of learning algorithms. We used the implementations available on the R software environment [R Core Team, 2018]. The modelling techniques selected were: Neural Networks (NNET), Multivariate Adaptive Regression Splines (MARS), Support Vector Machines (SVM) and Random Forests (RF). The learning algorithms, tested parameter variants, and corresponding R packages are shown in Table 5.2.

Table 5.2: Regression algorithms, parameter variants, and the respective R packages.

Learner	Parameter Variants	R package
NNET	$size = \{1, 2, 5, 10\}, decay = \{0, 0.01\}$	`nnet` [Venables and Ripley, 2002]
MARS	$nk = \{10, 17\}, degree = \{1, 2\}, thresh = \{0.01, 0.001\}$	`earth` [Milborrow, 2012]
SVM	$cost = \{10, 150, 300\}, gamma = \{0.01, 0.001\}$	`e1071` [Dimitriadou et al., 2011]
RF	$mtry = \{5, 7\}, ntree = \{500, 750, 1500\}$	`randomForest` [Liaw and Wiener, 2002]

We applied 28 learning approaches (8 NNET variants + 8 MARS variants + 6 SVM variants + 6 RF variants) to each of the 20 data sets, for a total of 560 setups.

Evaluation Methodology

The results were evaluated using a large set of metrics that are suitable for these problems, as discussed in Section 3.4 (page 93). Namely, we have used the following measures: F_1^{ϕ}, $G{-}Mean^{\phi}$, $prec^{\phi}$ rec^{ϕ}, $spec^{\phi}$ and $NPval^{\phi}$. However, given the extension of the experimental evaluation we will report here only the results for F_1^{ϕ} and $G - Mean^{\phi}$ measures. The full results for all metrics are included in Annex A. The values of the metrics were estimated by 2 repetitions of a 10-fold stratified cross validation procedure. All experiments were carried out in R software environment [R Core Team, 2018] and we used the stratified cross validation process implemented in the `performanceEstimation` R package [Torgo, 2014].

5.4.2 Evaluation of Unbiased Pre-processing Strategies

In this set of experiments we evaluated the impact in the performance of using unbiased pre-processing strategies. We compared the results of applying the 6 pre-processing strategies against the baseline of using the original imbalanced data set. More specifically, the following alternatives were compared: carry out no pre-processing (None), RU, RO, WERCS, GN, SMOTER and SMOGN. In these experiments all strategies were applied with the goal of balancing the training set. Only for WERCS strategy we were not able to guarantee that the final training set was balanced, given the characteristics of this method. Therefore, we fixed both *u.perc* and *o.perc* to 1. This choice was motivated by the following aspects:

i) it allows to obtain only one variant of the algorithm which provides a fair comparison against the other pre-processing strategies that were also only tested with one variant; and ii) by using the same value for both parameters the same importance is given to under and over-sampling in the WERCS algorithm. Table 5.3 summarises the tested variants of the unbiased pre-processing strategies.

Table 5.3: Tested variants of unbiased pre-processing strategies.

Strategy		**Parameter Variants**
None	No sampling applied	
RU	Random Under-sampling	$\%u.perc = \{balance\}$
RO	Random Over-sampling	$\%o.perc = \{balance\}$
WERCS	WEighted Relevance-based Combination Strategy	$\%u = \{1\}$, $\%o = \{1\}$
GN	Introduction of Gaussian Noise	$\%u.perc/\%o.perc = \{balance\}$, $\delta = \{0.01\}$
SMOTER	SMOTE for Regression	$\%u.perc/\%o.perc = \{balance\}$ $k = 5$
SMOGN	SMOTER with Gaussian Noise	$\%u.perc/\%o.perc = \{balance\}$, $k = 5$, $\delta = \{0.01\}$

Figures 5.12 and 5.13 show the results averaged over the learning algorithms variants for the F_1^ϕ and $G - Mean^\phi$ metrics. The complete set of results from these experiments are provided in Annex A.1.

The results show an advantage when applying pre-processing strategies for both F_1^ϕ and $G - Mean^\phi$ evaluation measures. Moreover, we are also able to confirm that the results obtained using the two evaluation measures are different. Let us observe, for instance, the behaviour of data set *a7*, when using the SVM learner. In this case, when considering the F_1^ϕ metric, we observe that applying random under-sampling improves the performance with respect to not applying any sampling (the F_1^ϕ score changes from 0.107 to 0.325). However, for the $G - Mean^\phi$ metric, the inverse happens: this metric score gets worse with the use of random under-sampling strategy (the $G - Mean^\phi$ score changes from 0.434 to 0.423). We also observe that globally, there is a higher positive impact when applying pre-processing strategies on the F_1^ϕ measure results in all learning algorithms in comparison to the results of $G - Mean^\phi$ measure. A possible explanation for this may be related with the metrics definition, as F_1^ϕ is focused in the performance on the rare cases, while $G - Mean^\phi$ takes into account the performance in both the normal and the rare cases. Still, for both measures, the use of pre-processing strategies shows a clear advantage with respect to using the original imbalanced data set.

We also detected two particular behaviours in some data sets that are noteworthy. One is related with data sets that already show a good performance when using the imbalanced data set. For these data sets, we observe that the use of pre-processing strategies has a marginal impact in the achieved performance. This is in accordance with what is expected because: i) when the baseline performance is already high there is a narrow margin for improvements; and ii) an initial good performance may indicate that the rare cases are easily separated from the normal cases and therefore the application of a pre-processing method has only a marginal

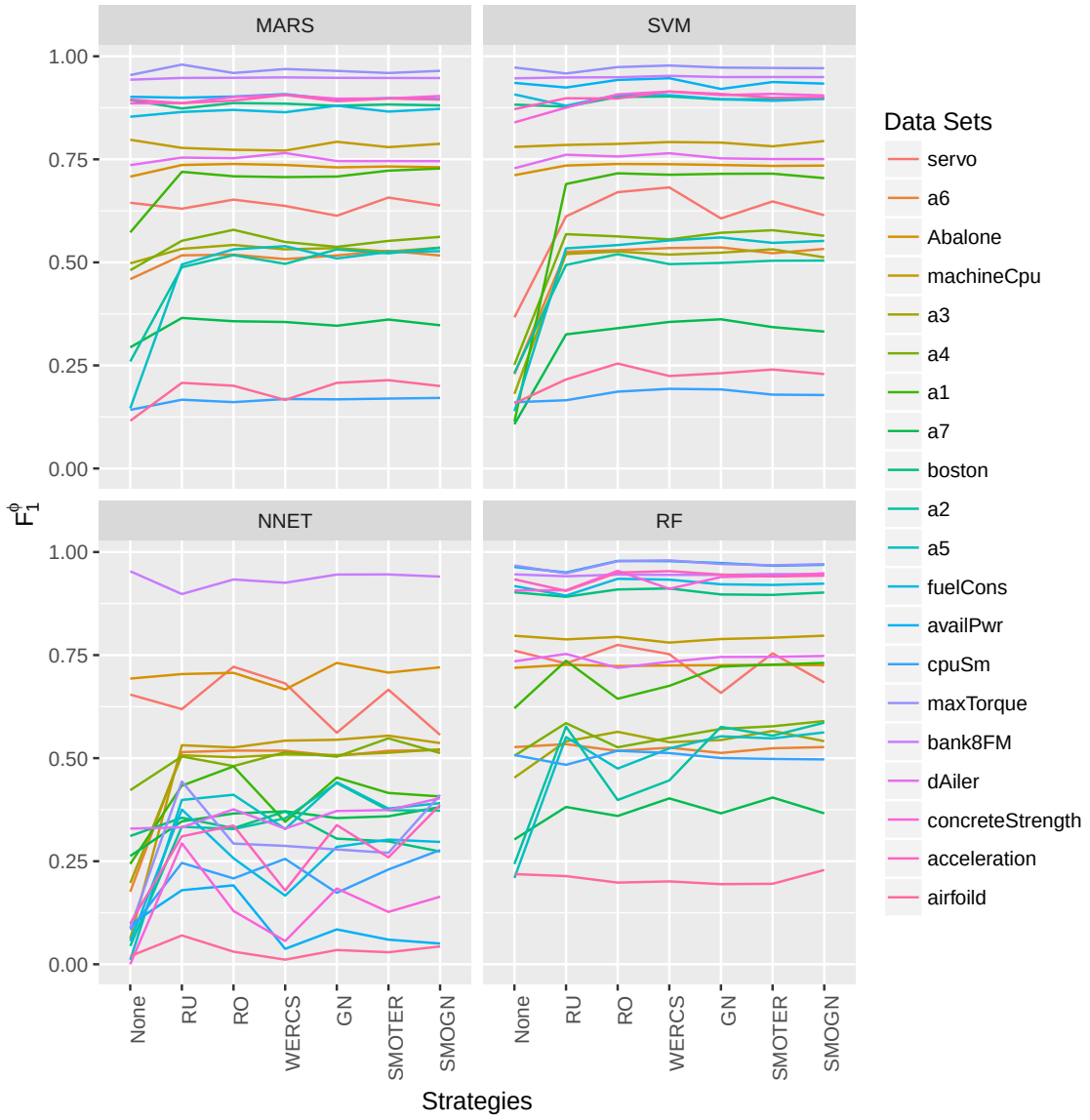


Figure 5.12: Average F_1^ϕ results of unbiased pre-processing strategies, by learning algorithm.

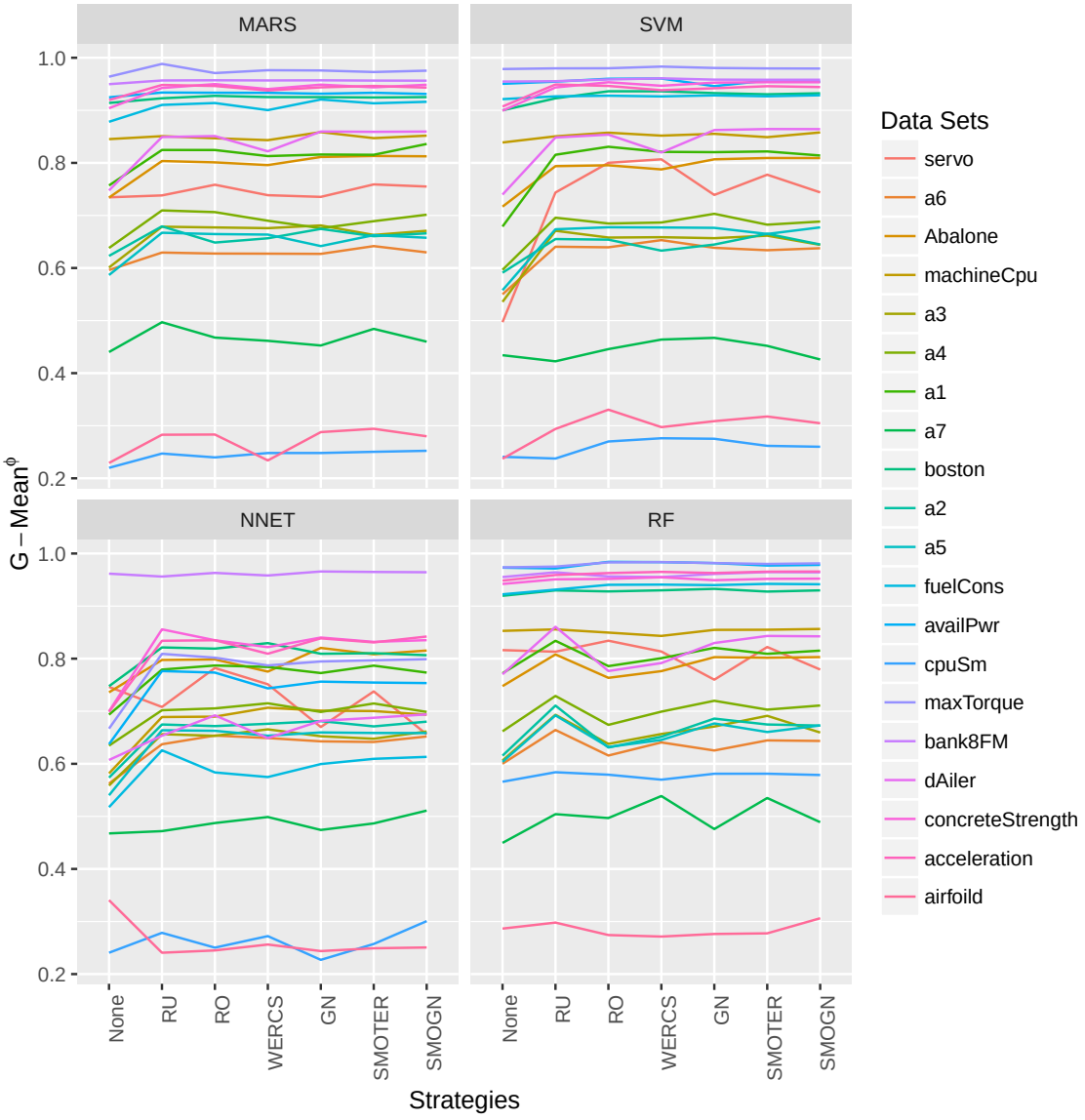


Figure 5.13: Average $G - Mean^{\phi}$ results of unbiased pre-processing strategies, by learning algorithm.

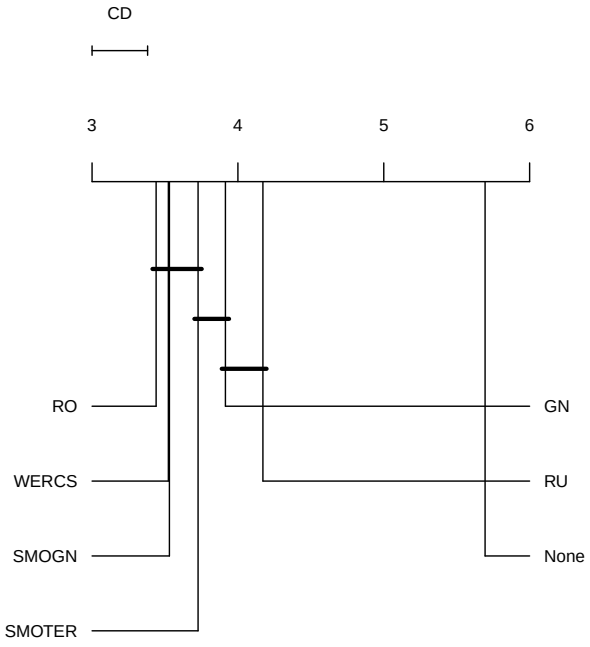
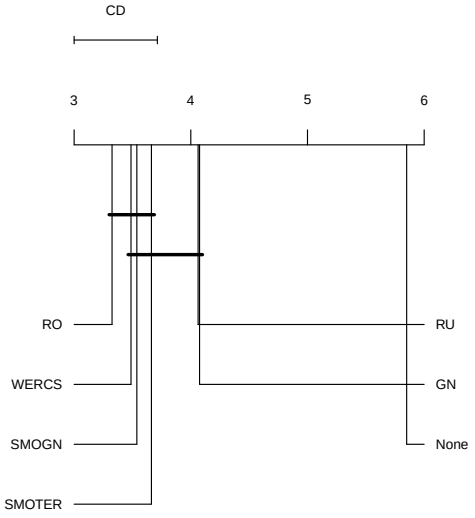


Figure 5.14: Critical Difference diagram of average F_1^ϕ results on all tested learners for unbiased pre-processing strategies.

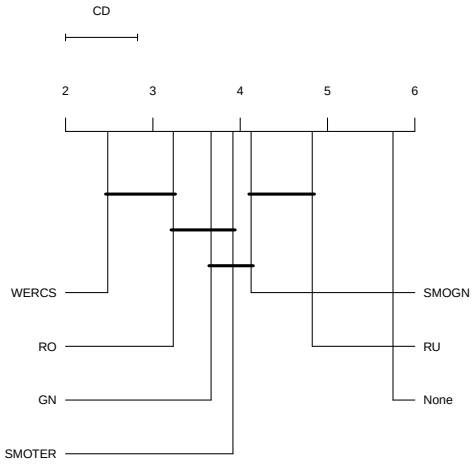
impact. The second observed behaviour is connected with a poor performance displayed on some data sets which is roughly maintained with the application of pre-processing strategies. We verify that this happens for both F_1^ϕ and $G - Mean^\phi$ measures, in a small fraction of the tested data sets. In these cases, the impact of the pre-processing strategies is also marginal. This problem may be related with some data characteristics that increase the problem difficulty.

To check the statistical significance of the observed differences, we applied the non-parametric Friedman F-test. This test considers the hypothesis that all approaches exhibit the same performance. If the test rejects this hypothesis, we proceed with the application of the post-hoc Nemenyi test for verifying which approaches are statistically different. We used CD diagrams [Demšar, 2006] to display this information and we set the significance level to 95%. Figure 5.14 displays the CD diagram of the aggregated F_1^ϕ over all tested learning algorithms variants. To provide a more detailed overview, we also show in Figure 5.15 the CD diagrams with the F_1^ϕ results by learner.

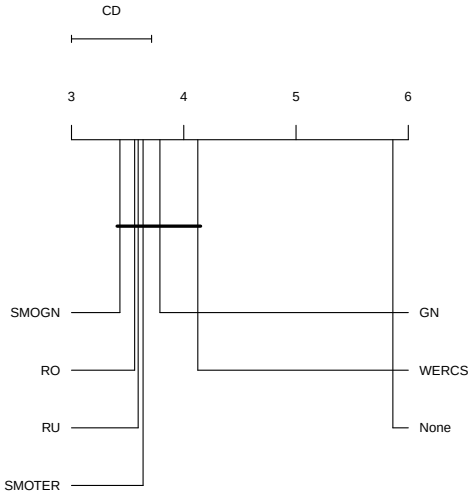
The aggregated results in Figure 5.14 confirm that the use of pre-processing strategies provides results that are better, with statistical significance, in comparison to using the original imbalanced training set. They also show that the differences between RO, WERCS, SMOGN and SMOTER are not statistically significant. This is also the case between SMOTER and GN, and between GN and RU. Regarding the results when considering each learner individually, show in Figure 5.15, we observe that the use of pre-processing strategies



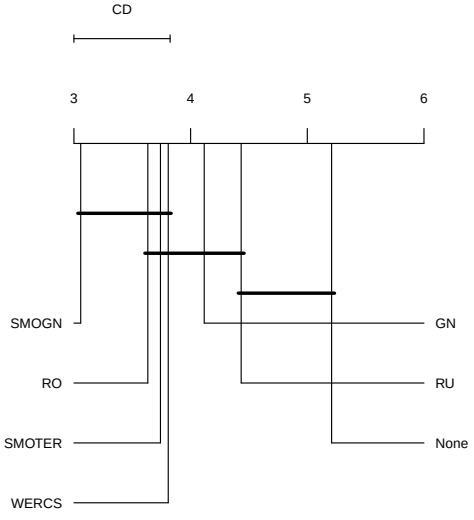
(a) Results of MARS.



(b) Results of SVM.



(c) Results of NNET.



(d) Results of RF.

Figure 5.15: Critical Difference diagrams of average F_1^{ϕ} results by learner for unbiased pre-processing strategies.

has a different impact depending on the learning algorithm. Still, we again observe that the use of the original training set is always worst than using the pre-processing approaches with statistical significance, except for the Random Forest learner where None and RU do not display a statistically significant difference. We also observe that SMOGN and RO obtain the best rankings for NNET and RF, while RO and WERCS achieve the best rankings for MARS and SVM.

5.4.3 Evaluation of Biased Pre-processing Strategies

In this section we describe the experiments with the biased pre-processing strategies. We will observe the impact in the performance of introducing a neighbourhood bias in a pre-processing strategy when compared to the use of the corresponding unbiased pre-processing strategy. We used the RU and the SMOTER strategies as baselines and compare them to the variants where we add a neighbourhood bias to these two approaches. Table 5.4 shows the considered variants for each base strategy and the respective used acronyms. All the tested pre-processing strategies aim at balancing the number of rare and normal cases.

Table 5.4: Summary of tested pre-processing variants with and without a neighbourhood bias.

Bias	Acronym	Base Strategy	Normal	Rare	Parameter Variants
Unbiased	U._._	RU	-	-	$\%u.perc = \{balance\}$
Biased	U.F._	RU	frontier	-	$\%u.perc = \{balance\}$
Biased	U.S._	RU	safe	-	$\%u.perc = \{balance\}$
Unbiased	S._._	SMOTER	-	-	$\%u.perc/\%o.perc = \{balance\}$ $k = 5$
Biased	S.F.F	SMOTER	frontier	frontier	$\%u.perc/\%o.perc = \{balance\}$ $k = 5$
Biased	S.F.S	SMOTER	frontier	safe	$\%u.perc/\%o.perc = \{balance\}$ $k = 5$
Biased	S.S.F	SMOTER	safe	frontier	$\%u.perc/\%o.perc = \{balance\}$ $k = 5$
Biased	S.S.S	SMOTER	safe	safe	$\%u.perc/\%o.perc = \{balance\}$ $k = 5$

The complete set of results regarding the biased pre-processing strategies for the different evaluation measures are provided in Annex A.2. Figures 5.16 and 5.17 show the average results of the tested learning algorithm variants for the F_1^ϕ metric on the under-sampling based and the SMOTER based strategies, respectively. In Table 5.5 we show the number of data sets for which a given pre-processing strategy displayed the best average F_1^ϕ score by learning algorithm. Finally, Figure 5.18 shows the over all number of data sets with best average F_1^ϕ scores by learning algorithm and type of pre-processing strategy.

The presented F_1^ϕ results show that the magnitude of the gains achieved when applying neighbourhood biased pre-processing strategies are not large. Still, we must highlight that frequently there are gains when applying a biased pre-processing strategy in comparing to using an unbiased pre-processing strategy. It is difficult to identify which type of bias is

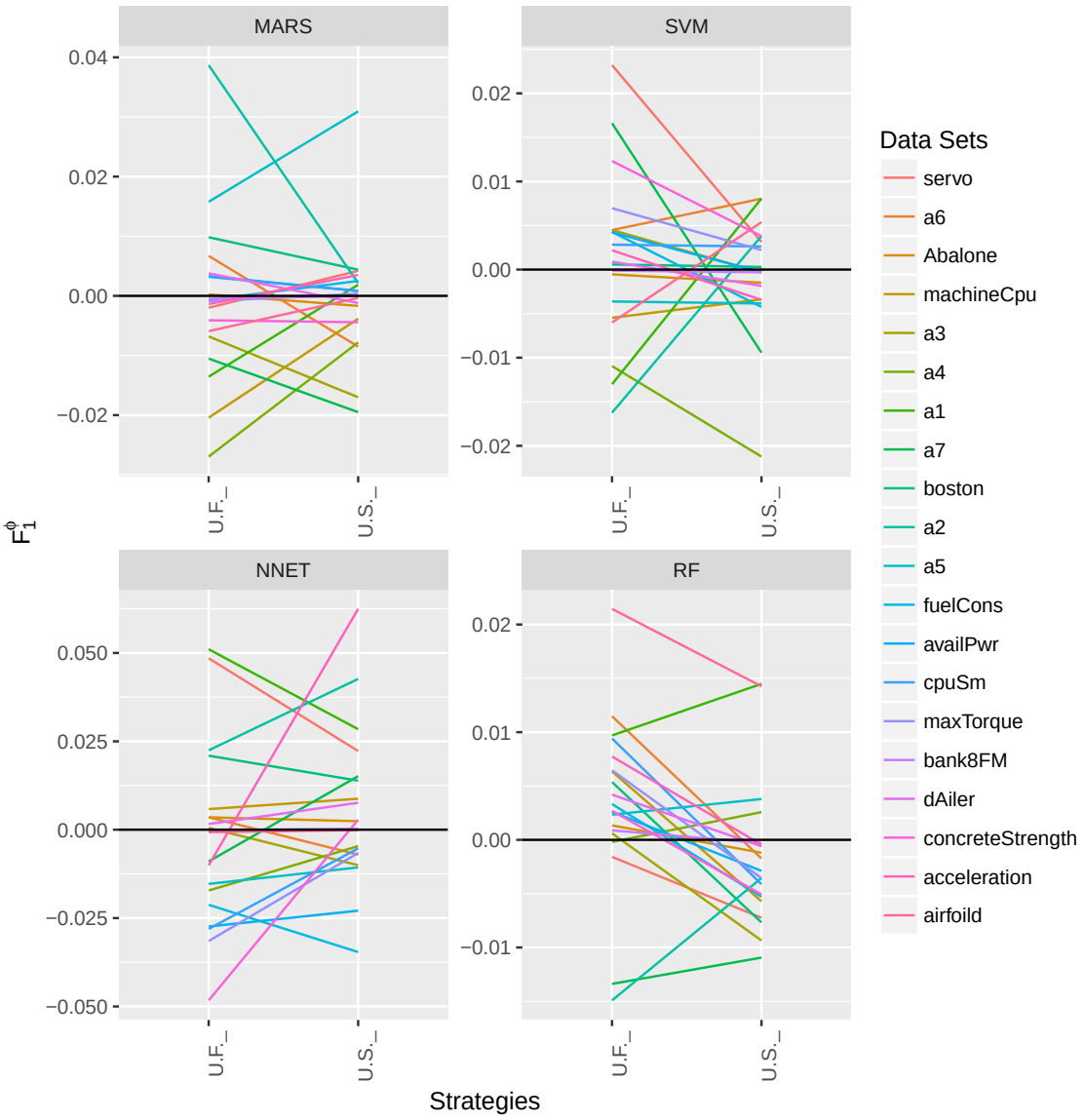


Figure 5.16: Average results of F_1^ϕ gains of under-sampling based biased pre-processing strategies in comparison to the RU strategy, for all tested learner variants.

Table 5.5: Number of data sets with best average F_1^ϕ score by learner and pre-processing strategy (biased and unbiased).

Learner	None	*U._._*	*U.F._*	*U.S._*	*S._._*	*S.F.F*	*S.S.F*	*S.F.S*	*S.S.S*
MARS	2	4	3	1	2	**6**	2	2	2
RF	3	0	4	1	4	1	3	**6**	0
SVM	2	1	1	0	2	**7**	2	5	5
NNET	1	**5**	3	3	3	0	2	4	1
Total	8	10	11	5	11	14	9	**17**	8

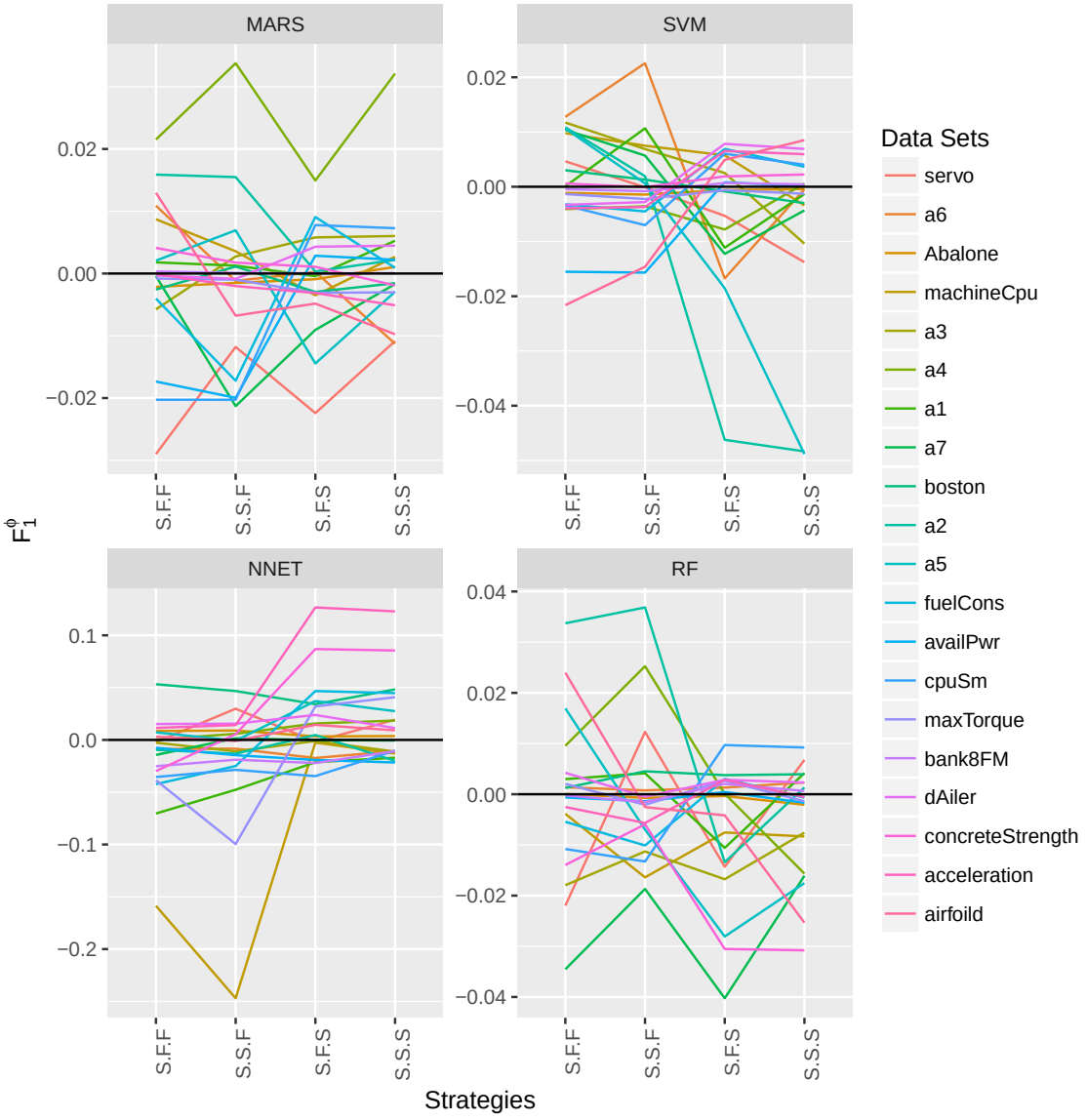


Figure 5.17: Average results of F_1^ϕ gains of SMOTER based biased pre-processing strategies in comparison to the SMOTER strategy, for all tested learner variants.

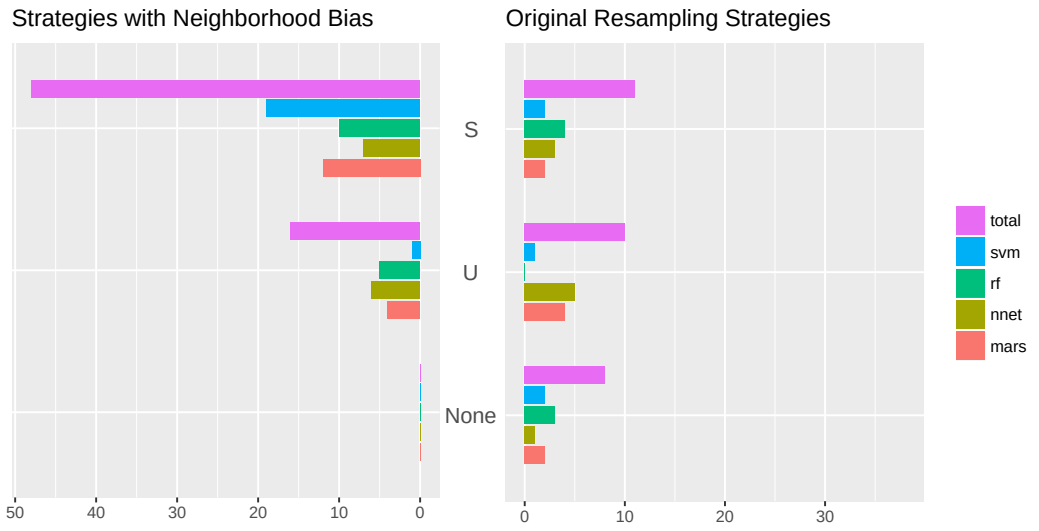


Figure 5.18: Number of data sets with best average F_1^ϕ scores by learner and strategy type for biased pre-processing strategies (S: SMOTER based; U: under-sampling based).

globally the best. Still, we observe that biased pre-processing strategies that reinforce the frontier of the normal cases, such as *U.F._*, *S.F.F* and *S.F.S*, have the best average F_1^ϕ score on a larger number of data sets. We also observe that the used learning algorithm and base pre-processing strategy have a high influence on the performance gains of the biased pre-processing strategies.

We applied the non-parametric Friedman F-test, to verify the statistical significance of the observed differences. These results allowed the rejection of the null hypothesis that all the tested pre-processing approaches for dealing with imbalanced regression show the same performance. We proceeded with the application of the post-hoc Nemenyi test to check which approaches are statistically significant for a significance level of 95%. Figure 5.19 shows the CD diagrams [Demšar, 2006] with the overall results of the tested learners. Figure 5.20 displays the CD diagrams with the results obtained for each learning algorithm.

The CD diagrams results allow to confirm that, globally, the use of the original data set is worst with statistical significance than applying a pre-processing strategy (biased or unbiased). We observe that there are differences on the statistical significance of the results achieved by the biased pre-processing strategies on the several tested learning algorithms. Namely, for the MARS learner there is no statistical significance among all tested strategies, while for the SVM learner the performance of *U._._* and *U.S._*, with statistical significance. It is also noticeable that, for the RF learner, the strategy *None* is not worse with statistical significance than several pre-processing strategies. This does not happen with the remaining tested learners.

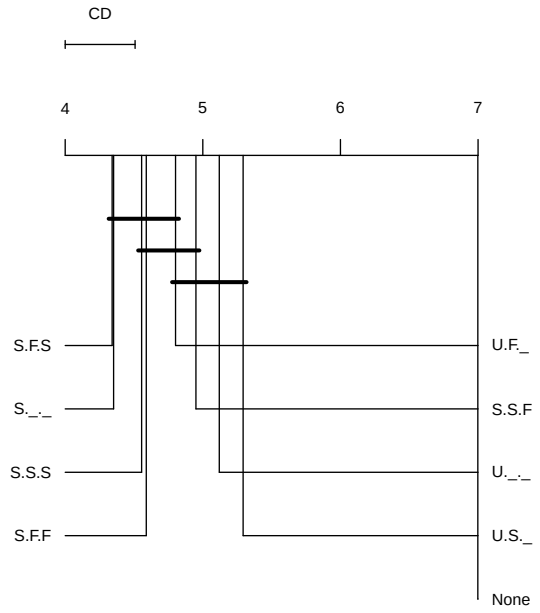
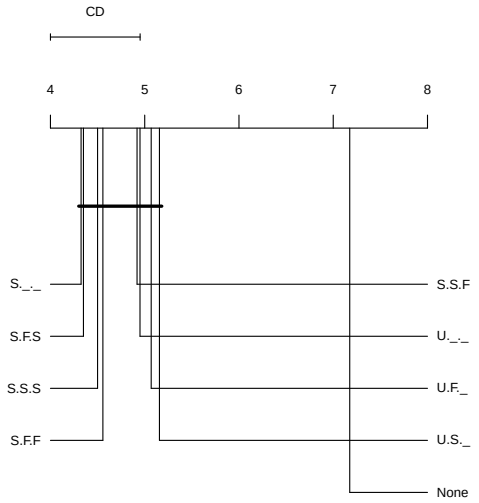


Figure 5.19: Critical Difference diagram of average F_1^{ϕ} results on all tested learners for biased pre-processing strategies.

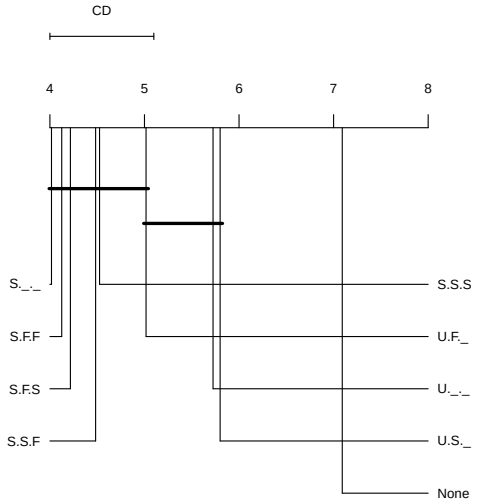
5.4.4 Evaluation of Different Distribution Changes on Pre-processing Strategies

In this section we report the experimental results of comparing the impact of applying pre-processing strategies with different parameters for changing the normal and rare cases distribution. We tested: i) the "balance" option which allows to roughly obtain the same number of rare and normal cases on the training set and ii) the application of other distribution modifications. The tested variants are described in Table 5.6. We must highlight that for the WERCS algorithm there is no "balance" option. For this reason, we considered the option that assigns 1 to both parameters $\%u$ and $\%o$ on this algorithm as the corresponding to the "balance" option.

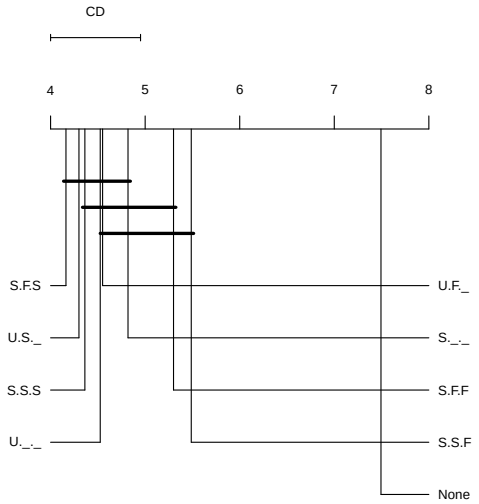
We changed the experimental setup used in the previous sets of experiments. We used the same 20 data sets and the same evaluation methodology. With respect to the learning algorithms, we have also used the same learners. However, we restricted the considered learner variants and only tested them with the default parameters as this approach is sufficient for answering our research question. The NNET learner requires the setting of parameter *size* which defines the number of nodes to use in the network hidden layer. We have set this parameter to 4. This choice was motivated by the recommendations of Cybenko [1989], Hornik [1991] for selecting a single hidden layer neural network. Blum [1992] suggests that the size of the hidden layer should be between the sizes of the input and output layers. This means that the size parameter should be between 1 and the number of features of the data set. For selecting a single number for the size parameter that satisfies this condition



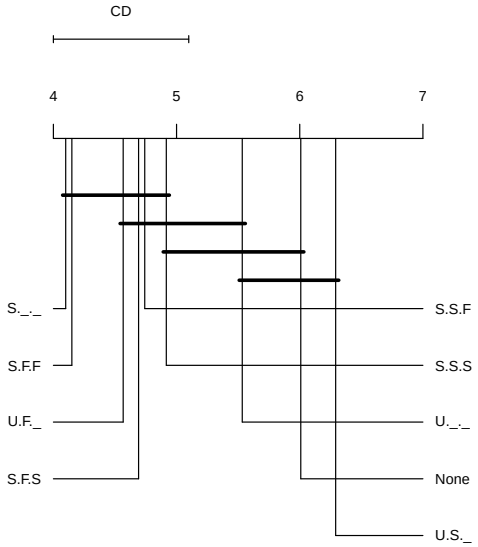
(a) Results of MARS.



(b) Results of SVM.



(c) Results of NNET.



(d) Results of RF.

Figure 5.20: Critical Difference diagrams of average F_1^{ϕ} results by learner for biased pre-processing strategies.

Table 5.6: Tested variants of pre-processing strategies with different distribution changes.

Strategy		**Parameter Variants**
None	No sampling applied (original data set)	
RU	Random Under-sampling	$\%u.perc = \{balance, 0.8, 0.6, 0.4\}$
RO	Random Over-sampling	$\%o.perc = \{balance, 1.5, 2, 2.5\}$
WERCS	WEighted Relevance-based Combination Strategy	$\%u = \{1, 2, 3\}$, $\%o = \{1, 2, 3\}$
GN	Introduction of Gaussian Noise	$\%u/\%o = \{balance, 0.8/1.5, 0.6/2, 0.4/2.5\}$, $\delta = \{0.01\}$
SMOTER	SMOTE for Regression	$\%u/\%o = \{balance, 0.8/1.5, 0.6/2, 0.4/2.5\}$, $k = 5$
SMOGN	SMOTER with Gaussian Noise	$\%u/\%o = \{balance, 0.8/1.5, 0.6/2, 0.4/2.5\}$, $k = 5$, $\delta = \{0.01\}$

for all data sets, we selected the value of 4 because the number of features in the used data sets varies between 4 and 37, therefore, the number 4 is suitable for all.

Figures 5.21 and 5.22 show the impact on the learners performance in two data sets (*boston* and *dAiler*[3]) when the pre-processing strategies aim at balancing the distribution of normal and rare cases or other changes in the distribution are applied. These figures display the average results of F_1^{ϕ} metric, by learner, for the strategy of balancing and the strategies that do not balance the normal and rare cases. The remaining results are available in Annex A.3.

The results on these data sets show that aiming at balancing the normal and rare cases is not always the best strategy. This was also observed in imbalanced classification tasks (e.g. Weiss and Provost [2003], Khoshgoftaar et al. [2007], Branco et al. [2016b]), and our results also support this conclusion for imbalanced regression problems.

5.5 Conclusions

The class imbalance problem has been studied for more than two decades. Still, only recently other predictive tasks also affected by this problem have started to receive some attention from the research community. This is the case with imbalanced regression problems. The problem of dealing with imbalanced regression tasks raises several different challenges. In this chapter we addressed the challenge of forcing the learning algorithms to focus on the most important cases in the context of imbalanced regression problems.

Several methods have been proposed to solve this challenge in a classification context. Among these methods we have the pre-processing strategies that change the original training set distribution. The goal of this change is to allow the learning algorithms to focus on the rare and important cases. In this chapter we proposed a diverse set of data pre-

[3]Data sets described in Section 5.4.1, Table 5.1 in page 142

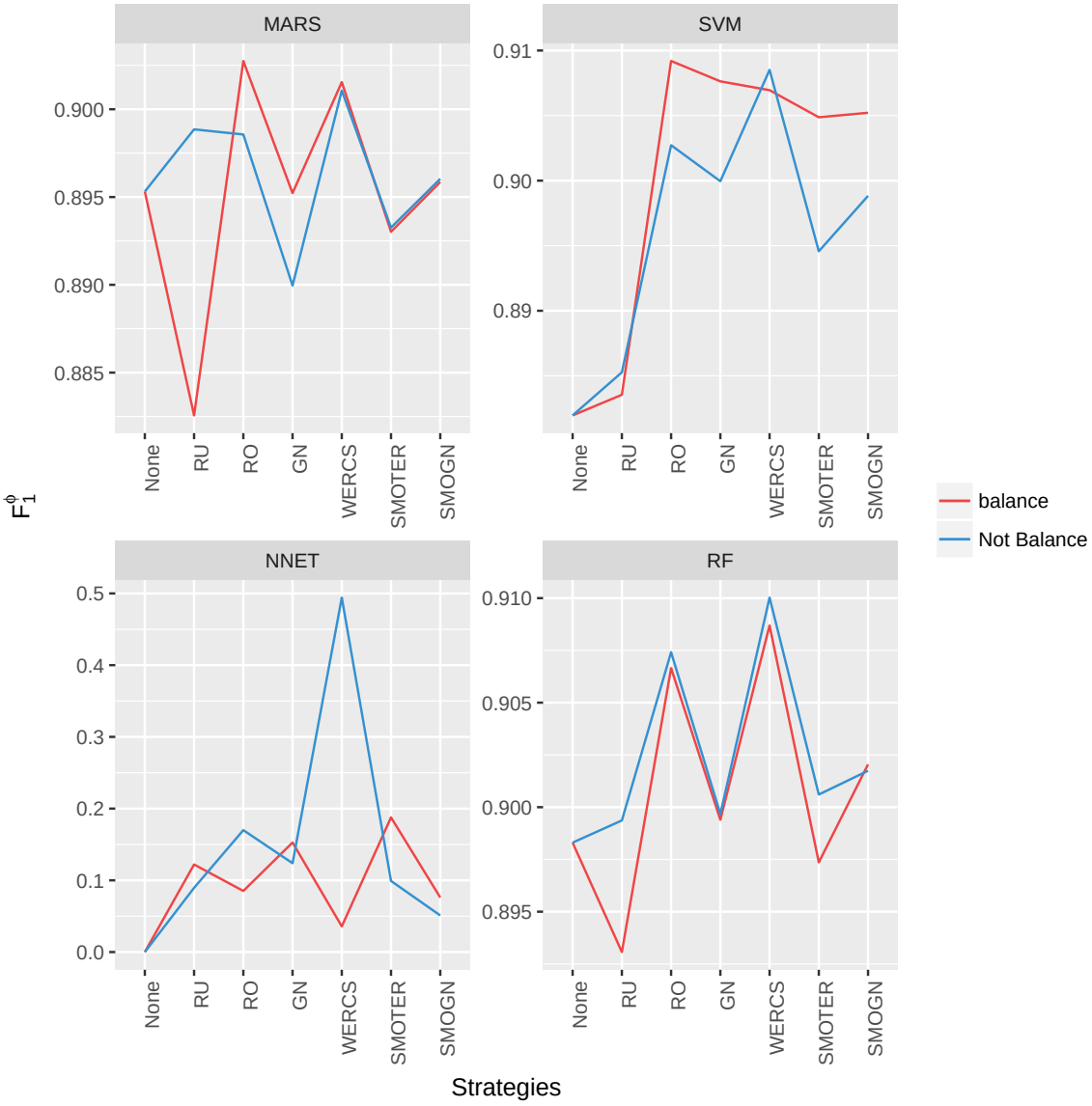


Figure 5.21: Results of F_1^ϕ measure on *boston* data set, by learner, with pre-processing strategies for balancing or considering other not balancing variants.

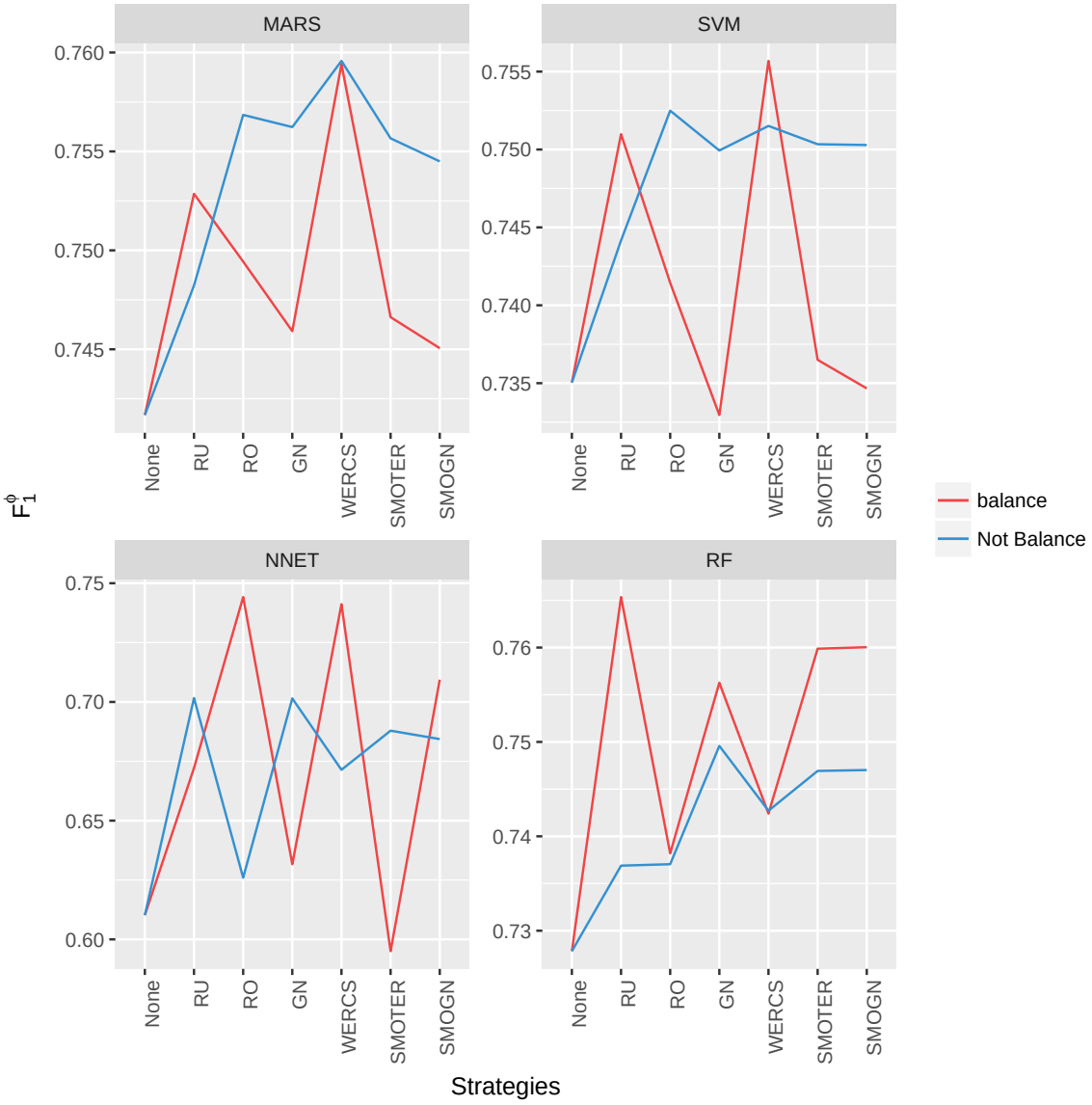


Figure 5.22: Results of F_1^ϕ measure on *dAiler* data set, by learner, with pre-processing strategies for balancing or considering other not balancing variants.

processing solutions to solve the imbalanced regression problem. An experimental study was carried out to assess the effectiveness of the proposed strategies. The main goals of the experimental analysis were: i) to understand the impact of applying pre-processing strategies in the learning algorithms performance; ii) observe if the use of neighbourhood biased pre-processing strategies brought advantages in relation to unbiased pre-processing strategies; and iii) exploring the impact of balancing the rare and normal cases in comparison to the application of other changes in the distribution. Regarding our first objective, our experiments showed a clear advantage in using data pre-processing strategies for dealing with imbalanced regression problems. Concerning the second research question, results show that biasing the pre-processing strategies by considering the cases neighbourhood has gains in comparison to the use of unbiased pre-processing strategies. However, it is not straightforward to decide which type of bias should be applied because we could not find one biased method that was the best overall. As for our third objective, the results showed that balancing the rare and normal cases is not always the best way for changing the examples distribution. These results were verified for different metrics and are in accordance with results obtained by others in the context of imbalanced classification problems.

Chapter 6

Conclusions

Utility-based predictive analytics is very important in several real world applications. These problems are characterised by the existence of information regarding non-uniform preferences over the domain of the target variable. This information on relevant domain regions provided is by the end-user as it is domain-dependent, and it describes important situations that may be associated with either high benefits or substantial costs. This is a very frequent problem occurring in a diversity of domains, such as medical, financial, or meteorological, among other. Moreover, the utility-based learning problem is also considered one of the most important problems that machine learning and data mining researchers need to address [Yang and Wu, 2006].

In a utility-based learning setting, it is crucial to properly use the information regarding the end-user preferences over the target domain. Still, important challenges arise when incorporating this knowledge in the learning procedure. The initial research on cost-sensitive classification provided a more solid background for classification tasks dealing with costs. Therefore, the current challenges regarding utility-based learning are more focused on regression problems. In general, the main open issues within utility-based learning are: i) the definition of a unifying framework for utility-based learning problems, ii) the definition of suitable performance assessment measures for utility-based learning tasks and their sub-problems, iii) the development of user-friendly methods for expressing the existing domain preferences, iv) the development of learning methods for utility optimisation, specially in a regression context, and v) the proposal of new methods to deal with the imbalanced regression sub-problem. In this thesis we have studied the problem of utility-based learning with a special focus on the open issues mentioned above.

6.1 Contributions

This thesis includes the following main contributions:

- **Literature Review of the Utility-Based Learning Problem.** In Chapter 2 we provided an extensive literature review of the utility-based learning problem and the sub-problem of learning from imbalanced domains. We included the theoretical developments, performance evaluation measures and learning methods for both classification and regression tasks. We also presented and discussed the main open challenges of utility-based learning.

- **A unifying framework for utility-based learning tasks.** In Chapter 3 we presented a unifying framework that contextualises utility-based learning tasks within standard and non-standard predictive tasks. Using this framework we showed the relationship between utility-based learning problems and other predictive tasks, such as standard tasks and the problem of learning from imbalanced domains.

- **Utility Surfaces through Spatial Interpolation.** In Chapter 3 we provided a solution for obtaining the full utility information in regression tasks. The main goal of this proposal was to provide a user-friendly tool for obtaining a complete utility surface with the minimum user effort using a selected spatial interpolation technique.

- **New Performance Evaluation Measures.** In Chapter 3 we proposed new measures for performance evaluation adjusted to the available problem information. These new measures are suitable for contexts where there is only a partial or informal information regarding the user preferences and allow to fully use the available relevant domain knowledge.

- **Methods for Utility Maximisation in Regression.** In Chapter 4 we proposed and evaluated two general methods for utility optimisation in regression tasks. Both methods present good results. We present the main advantages and disadvantages of the proposed methods and discuss the practical contexts where one method may be preferred over the other.

- **Pre-processing Methods for Dealing with Imbalanced Regression Problems.** In Chapter 5 we presented several data pre-processing methods for dealing with imbalanced regression problems. We explore methods that simply remove or add replicas of existing cases and methods that generate new synthetic cases. We also explore methods that take into account the cases neighbourhood or that disregard this information.

- **UBL R package.** An important outcome of this thesis is the UBL R package which contains the methods proposed in this thesis and several other methods described in the current literature for dealing with utility-based problems. The UBL package is open source software and is freely available to all research community.

6.2 Future Research Directions

The utility-based learning paradigm is still recent, specially in what concerns its application to regression tasks. The research and the results that we have presented in this thesis can be extended in several different directions. The following list presents some of those directions.

- **Performance Evaluation Procedures and Measures.** This is an important issue that still needs to be further addressed. Although several measures were proposed for utility-based learning and the sub-problem of learning from imbalanced domains, it is still necessary to explore new measures that can easily accommodate different levels and types of information concerning the user preference biases in a predictive task. It is also important to explore the impact of the used performance estimation procedures. For instance, it would be interesting to determine how small changes in this procedure impact the performance estimation.

- **Automatic Ways for Obtaining the Utility Information.** As we have seen, it is hard, from an end-user perspective, to provide the complete utility information for a certain problem, specially for regression tasks. It is necessary to develop new user-friendly ways for automatically converting the domain knowledge into formal information. Several paths may be considered, such as an active learning based procedure, or a mathematical formulation that is able to adapt to different settings.

- **Relationship between Data Characteristics and Performance.** Regarding the sub-problem of learning from imbalanced domains, it would be interesting to see if there is a relationship between data characteristics and the performance gains achieved by the different pre-processing methods. This could lead to a new recommendation system that could considerably reduce the end-user time when solving these tasks.

- **New Automatic Pre-processing Methods.** The developments accomplished so far still require a considerable effort from the end-user, namely when dealing with imbalanced domains. It would be interesting to develop an automatic method that could "learn", for a given problem, the best pre-processing strategy to apply.

- **Extension of Utility-based Learning Methods to Other Predictive Problems.** Several other prediction problems involve the consideration of costs and benefits which may be provided formally or informally. These are important problems which still require several improvements. Examples of such tasks include imbalanced data streams or multi-label classification problems.

- **Utility-based Learning with Instance or/and Spatio-temporal Dependant Costs and Benefits.** The issue of considering instance dependant costs and benefits was already addressed in a classification context. However, it is still necessary to

develop new strategies that are capable of dealing with more complex settings which may involve instance and/or spatio-temporal dependant utility.

Appendices

Appendix A

Results of the Experiments with Pre-processing Strategies for Tackling Imbalanced Regression Problems

In this appendix we present the complete set of obtained results from the experiments conducted in Chapter 5 regarding the use of pre-processing strategies for dealing with imbalanced regression problems. In Section A.1 we present the results of the unbiased strategies using multiple evaluation metrics. Section A.2 displays the complete results of the biased pre-processing strategies, and in Section A.3 the results of strategies with different changes on the training set distribution are presented.

A.1 Evaluation Results of Unbiased Pre-processing Strategies

Table A.1: Evaluation results of unbiased pre-processing strategies concerning the F_1^ϕ metric (average and standard deviation) for 20 data sets and all tested variants of 4 learners.

Learner	Data sets	None	RU	RO	WERCS	SMOTER	GN	SMOGN
MARS	servo	0.645(0.11)	0.630(0.13)	0.652(0.12)	0.637(0.12)	0.657(0.12)	0.613(0.14)	0.638(0.14)
	a6	0.459(0.19)	0.517(0.10)	0.519(0.09)	0.508(0.09)	0.528(0.10)	0.517(0.10)	0.517(0.09)
	Abalone	0.708(0.02)	0.736(0.01)	0.739(0.02)	0.736(0.02)	0.733(0.01)	0.730(0.01)	0.731(0.01)
	machineCpu	0.797(0.09)	0.778(0.08)	0.773(0.11)	0.771(0.09)	0.780(0.09)	0.793(0.08)	0.788(0.08)
	a3	0.498(0.21)	0.533(0.09)	0.542(0.10)	0.532(0.09)	0.526(0.09)	0.535(0.09)	0.536(0.10)
	a4	0.481(0.20)	0.552(0.11)	0.579(0.13)	0.549(0.13)	0.552(0.10)	0.538(0.12)	0.562(0.10)
	a1	0.573(0.25)	0.720(0.10)	0.709(0.09)	0.707(0.08)	0.722(0.10)	0.708(0.09)	0.728(0.09)
	a7	0.294(0.17)	0.365(0.13)	0.357(0.13)	0.355(0.13)	0.361(0.14)	0.346(0.14)	0.347(0.16)
	boston	0.894(0.04)	0.873(0.05)	0.887(0.04)	0.885(0.05)	0.883(0.04)	0.880(0.04)	0.881(0.04)
	a2	0.260(0.38)	0.488(0.22)	0.518(0.22)	0.496(0.23)	0.522(0.20)	0.531(0.21)	0.535(0.21)
	a5	0.146(0.21)	0.495(0.22)	0.532(0.20)	0.539(0.21)	0.525(0.21)	0.510(0.21)	0.527(0.20)
	fuelCons	0.853(0.03)	0.865(0.03)	0.870(0.02)	0.864(0.03)	0.866(0.03)	0.880(0.02)	0.872(0.03)
	availPwr	0.902(0.02)	0.900(0.02)	0.902(0.02)	0.908(0.02)	0.898(0.02)	0.894(0.02)	0.896(0.02)
	cpuSm	0.142(0.03)	0.167(0.02)	0.161(0.03)	0.169(0.03)	0.170(0.02)	0.168(0.02)	0.171(0.02)
	maxTorque	0.954(0.01)	0.980(0.01)	0.959(0.01)	0.969(0.01)	0.959(0.02)	0.965(0.02)	0.965(0.02)
	bank8FM	0.943(0.01)	0.947(0.01)	0.948(0.01)	0.949(0.01)	0.947(0.01)	0.948(0.01)	0.947(0.01)
	dAiler	0.736(0.03)	0.754(0.02)	0.753(0.02)	0.765(0.02)	0.746(0.02)	0.746(0.02)	0.746(0.02)
	concreteStrength	0.886(0.03)	0.886(0.03)	0.901(0.03)	0.906(0.03)	0.898(0.03)	0.897(0.03)	0.903(0.03)
	acceleration	0.895(0.03)	0.886(0.04)	0.892(0.03)	0.906(0.03)	0.898(0.03)	0.891(0.03)	0.895(0.02)
	airfoild	0.116(0.08)	0.208(0.09)	0.201(0.11)	0.166(0.11)	0.214(0.11)	0.208(0.10)	0.200(0.10)
RF	servo	0.761(0.14)	0.730(0.12)	0.775(0.14)	0.752(0.16)	0.754(0.15)	0.658(0.16)	0.683(0.14)
	a6	0.527(0.14)	0.534(0.12)	0.518(0.14)	0.525(0.13)	0.524(0.14)	0.513(0.11)	0.527(0.12)
	Abalone	0.719(0.02)	0.726(0.02)	0.723(0.02)	0.725(0.02)	0.727(0.02)	0.726(0.02)	0.725(0.01)
	machineCpu	0.797(0.09)	0.788(0.09)	0.794(0.10)	0.780(0.10)	0.792(0.09)	0.789(0.08)	0.797(0.09)
	a3	0.453(0.20)	0.541(0.09)	0.564(0.15)	0.539(0.12)	0.566(0.09)	0.544(0.09)	0.541(0.10)

Continued on next page

Table A.1 – continued from previous page

Learner	Data sets	None	RU	RO	WERCS	SMOTER	GN	SMOGN
	a4	0.506(0.24)	0.585(0.12)	0.526(0.23)	0.549(0.16)	0.577(0.13)	0.571(0.16)	0.590(0.14)
	a1	0.621(0.29)	0.736(0.08)	0.644(0.24)	0.675(0.18)	0.727(0.09)	0.722(0.10)	0.731(0.10)
	a7	0.303(0.19)	0.382(0.15)	0.360(0.19)	0.402(0.17)	0.404(0.15)	0.366(0.17)	0.366(0.17)
	boston	0.902(0.04)	0.891(0.05)	0.909(0.04)	0.912(0.04)	0.896(0.04)	0.897(0.05)	0.902(0.04)
	a2	0.243(0.36)	0.576(0.22)	0.398(0.31)	0.446(0.31)	0.555(0.23)	0.576(0.27)	0.586(0.26)
	a5	0.209(0.30)	0.551(0.20)	0.475(0.28)	0.523(0.24)	0.548(0.22)	0.553(0.22)	0.563(0.22)
	fuelCons	0.918(0.01)	0.894(0.02)	0.935(0.01)	0.933(0.02)	0.920(0.02)	0.922(0.02)	0.923(0.02)
RF	availPwr	0.964(0.02)	0.951(0.02)	0.978(0.01)	0.978(0.02)	0.967(0.01)	0.973(0.02)	0.969(0.01)
	cpuSm	0.508(0.05)	0.484(0.06)	0.518(0.05)	0.512(0.06)	0.498(0.06)	0.500(0.06)	0.497(0.06)
	maxTorque	0.967(0.01)	0.949(0.02)	0.978(0.01)	0.980(0.01)	0.968(0.01)	0.971(0.01)	0.971(0.01)
	bank8FM	0.946(0.01)	0.941(0.01)	0.945(0.01)	0.945(0.01)	0.946(0.01)	0.945(0.01)	0.946(0.01)
	dAiler	0.735(0.03)	0.753(0.01)	0.719(0.02)	0.734(0.03)	0.746(0.02)	0.746(0.02)	0.748(0.02)
	concreteStrength	0.907(0.21)	0.907(0.03)	0.954(0.02)	0.911(0.22)	0.942(0.03)	0.939(0.03)	0.948(0.03)
	acceleration	0.934(0.02)	0.906(0.02)	0.950(0.02)	0.954(0.01)	0.941(0.02)	0.945(0.02)	0.943(0.02)
	airfoild	0.219(0.15)	0.214(0.09)	0.198(0.09)	0.201(0.09)	0.195(0.10)	0.194(0.11)	0.229(0.10)
	servo	0.366(0.12)	0.612(0.10)	0.670(0.10)	0.682(0.09)	0.648(0.10)	0.607(0.11)	0.614(0.11)
	a6	0.229(0.18)	0.526(0.11)	0.530(0.10)	0.535(0.11)	0.522(0.10)	0.536(0.10)	0.533(0.10)
	Abalone	0.712(0.03)	0.735(0.02)	0.739(0.02)	0.738(0.02)	0.734(0.01)	0.736(0.01)	0.735(0.01)
	machineCpu	0.780(0.10)	0.785(0.07)	0.787(0.08)	0.792(0.08)	0.782(0.08)	0.791(0.08)	0.794(0.07)
	a3	0.181(0.10)	0.520(0.10)	0.526(0.08)	0.519(0.09)	0.532(0.10)	0.524(0.08)	0.512(0.10)
	a4	0.252(0.24)	0.568(0.10)	0.563(0.10)	0.556(0.09)	0.578(0.12)	0.572(0.12)	0.564(0.11)
	a1	0.113(0.18)	0.690(0.08)	0.716(0.08)	0.713(0.07)	0.715(0.09)	0.715(0.09)	0.704(0.10)
SVM	a7	0.107(0.10)	0.325(0.16)	0.340(0.14)	0.356(0.15)	0.343(0.15)	0.362(0.16)	0.332(0.15)
	boston	0.883(0.04)	0.878(0.04)	0.901(0.03)	0.903(0.04)	0.896(0.04)	0.895(0.04)	0.898(0.04)
	a2	0.232(0.34)	0.494(0.22)	0.520(0.22)	0.496(0.25)	0.504(0.24)	0.499(0.24)	0.504(0.24)
	a5	0.139(0.17)	0.534(0.21)	0.542(0.23)	0.553(0.22)	0.547(0.22)	0.560(0.22)	0.552(0.21)
	fuelCons	0.908(0.02)	0.880(0.03)	0.906(0.02)	0.906(0.03)	0.892(0.02)	0.896(0.02)	0.897(0.02)
	availPwr	0.935(0.01)	0.924(0.02)	0.943(0.01)	0.947(0.01)	0.938(0.01)	0.921(0.02)	0.934(0.01)
	cpuSm	0.161(0.03)	0.166(0.03)	0.186(0.03)	0.193(0.03)	0.179(0.03)	0.192(0.03)	0.178(0.03)
	maxTorque	0.973(0.01)	0.958(0.02)	0.974(0.01)	0.978(0.01)	0.972(0.01)	0.973(0.01)	0.971(0.01)

Continued on next page

Table A.1 – continued from previous page

Learner	Data sets	None	RU	RO	WERCS	SMOTER	GN	SMOGN
SVM	bank8FM	0.947(0.01)	0.949(0.01)	0.949(0.01)	0.952(0.01)	0.950(0.01)	0.950(0.01)	0.950(0.01)
	dAiler	0.728(0.03)	0.761(0.02)	0.757(0.02)	0.765(0.02)	0.751(0.01)	0.752(0.01)	0.751(0.01)
	concreteStrength	0.840(0.14)	0.875(0.04)	0.908(0.03)	0.914(0.04)	0.909(0.03)	0.906(0.03)	0.905(0.03)
	acceleration	0.872(0.07)	0.898(0.03)	0.897(0.02)	0.914(0.02)	0.899(0.02)	0.909(0.02)	0.901(0.02)
	airfoild	0.158(0.09)	0.216(0.12)	0.255(0.13)	0.224(0.12)	0.240(0.11)	0.231(0.11)	0.229(0.11)
NNET	servo	0.654(0.14)	0.619(0.18)	0.722(0.17)	0.681(0.18)	0.666(0.18)	0.562(0.17)	0.556(0.17)
	a6	0.176(0.19)	0.515(0.11)	0.518(0.11)	0.518(0.11)	0.518(0.11)	0.505(0.13)	0.520(0.11)
	Abalone	0.693(0.06)	0.704(0.08)	0.707(0.11)	0.667(0.10)	0.708(0.08)	0.731(0.01)	0.720(0.05)
	machineCpu	0.062(0.08)	0.531(0.10)	0.526(0.10)	0.542(0.09)	0.555(0.09)	0.545(0.09)	0.537(0.11)
	a3	0.197(0.21)	0.507(0.08)	0.502(0.08)	0.509(0.07)	0.513(0.08)	0.508(0.08)	0.521(0.08)
	a4	0.422(0.25)	0.504(0.15)	0.481(0.19)	0.514(0.15)	0.548(0.14)	0.504(0.17)	0.512(0.13)
	a1	0.243(0.26)	0.433(0.24)	0.480(0.22)	0.345(0.27)	0.416(0.26)	0.453(0.28)	0.407(0.29)
	a7	0.263(0.23)	0.347(0.14)	0.366(0.16)	0.371(0.15)	0.359(0.14)	0.355(0.14)	0.381(0.15)
	boston	0.311(0.29)	0.355(0.21)	0.330(0.27)	0.371(0.29)	0.298(0.24)	0.305(0.24)	0.274(0.27)
	a2	0.044(0.13)	0.334(0.22)	0.328(0.26)	0.354(0.27)	0.374(0.26)	0.440(0.27)	0.373(0.28)
	a5	0.011(0.04)	0.399(0.23)	0.411(0.25)	0.328(0.27)	0.378(0.24)	0.442(0.22)	0.391(0.24)
	fuelCons	0.056(0.13)	0.375(0.20)	0.257(0.20)	0.166(0.20)	0.302(0.24)	0.285(0.18)	0.297(0.22)
	availPwr	0.096(0.12)	0.180(0.21)	0.192(0.29)	0.037(0.09)	0.060(0.14)	0.085(0.18)	0.050(0.10)
	cpuSm	0.084(0.07)	0.246(0.17)	0.208(0.15)	0.256(0.23)	0.230(0.17)	0.173(0.17)	0.277(0.21)
	maxTorque	0.088(0.17)	0.443(0.22)	0.293(0.25)	0.287(0.26)	0.270(0.18)	0.278(0.30)	0.411(0.26)
	bank8FM	0.953(0.01)	0.898(0.10)	0.933(0.05)	0.925(0.08)	0.946(0.03)	0.945(0.03)	0.940(0.06)
	dAiler	0.329(0.08)	0.332(0.03)	0.376(0.12)	0.329(0.05)	0.374(0.06)	0.372(0.11)	0.403(0.08)
	concreteStrength	0.000(0.00)	0.293(0.23)	0.129(0.24)	0.056(0.12)	0.127(0.19)	0.184(0.19)	0.164(0.17)
	acceleration	0.099(0.24)	0.310(0.22)	0.336(0.31)	0.179(0.25)	0.259(0.25)	0.338(0.27)	0.387(0.28)
	airfoild	0.020(0.03)	0.070(0.06)	0.031(0.05)	0.011(0.02)	0.029(0.04)	0.035(0.04)	0.043(0.06)

Table A.2: Evaluation results concerning the $G-Mean^{\phi}$ metric (average and standard deviation) for 20 data sets and all tested variants of 4 learners using unbiased pre-processing strategies.

Learner	Data sets	None	RU	RO	WERCS	SMOTER	GN	SMOGN
MARS	servo	0.734(0.12)	0.738(0.12)	0.758(0.12)	0.739(0.12)	0.759(0.12)	0.735(0.12)	0.755(0.13)
	a6	0.596(0.11)	0.629(0.14)	0.628(0.12)	0.627(0.13)	0.642(0.13)	0.627(0.14)	0.630(0.13)
	Abalone	0.734(0.02)	0.803(0.02)	0.801(0.02)	0.796(0.02)	0.813(0.02)	0.811(0.02)	0.813(0.02)
	machineCpu	0.845(0.07)	0.851(0.07)	0.847(0.08)	0.843(0.06)	0.847(0.08)	0.859(0.06)	0.852(0.07)
	a3	0.601(0.12)	0.679(0.11)	0.677(0.13)	0.676(0.11)	0.663(0.10)	0.681(0.11)	0.671(0.12)
	a4	0.638(0.11)	0.710(0.14)	0.706(0.15)	0.690(0.14)	0.689(0.12)	0.676(0.14)	0.701(0.12)
	a1	0.757(0.10)	0.825(0.09)	0.824(0.09)	0.813(0.09)	0.815(0.09)	0.816(0.09)	0.836(0.09)
	a7	0.440(0.18)	0.497(0.20)	0.468(0.20)	0.462(0.18)	0.484(0.21)	0.453(0.20)	0.460(0.23)
	boston	0.914(0.05)	0.923(0.04)	0.927(0.04)	0.925(0.05)	0.924(0.04)	0.925(0.04)	0.925(0.04)
	a2	0.623(0.25)	0.679(0.25)	0.648(0.26)	0.657(0.26)	0.660(0.25)	0.674(0.26)	0.666(0.26)
	a5	0.587(0.23)	0.667(0.26)	0.665(0.25)	0.664(0.26)	0.663(0.26)	0.642(0.27)	0.658(0.26)
	fuelCons	0.878(0.02)	0.910(0.02)	0.914(0.01)	0.900(0.02)	0.913(0.02)	0.921(0.02)	0.916(0.02)
	availPwr	0.925(0.02)	0.934(0.01)	0.933(0.01)	0.933(0.01)	0.933(0.01)	0.932(0.01)	0.931(0.01)
	cpuSm	0.220(0.03)	0.247(0.03)	0.240(0.04)	0.248(0.04)	0.250(0.03)	0.248(0.03)	0.252(0.03)
	maxTorque	0.964(0.01)	0.988(0.00)	0.971(0.01)	0.976(0.01)	0.973(0.01)	0.976(0.01)	0.975(0.01)
	bank8FM	0.950(0.01)	0.957(0.01)	0.957(0.01)	0.957(0.01)	0.957(0.01)	0.957(0.01)	0.956(0.01)
	dAiler	0.748(0.02)	0.849(0.02)	0.851(0.02)	0.822(0.02)	0.859(0.02)	0.860(0.02)	0.860(0.02)
	concreteStrength	0.904(0.03)	0.942(0.03)	0.950(0.02)	0.940(0.03)	0.944(0.03)	0.949(0.02)	0.949(0.02)
	acceleration	0.919(0.02)	0.948(0.02)	0.946(0.01)	0.938(0.02)	0.946(0.02)	0.944(0.02)	0.943(0.02)
	airfoild	0.229(0.07)	0.283(0.10)	0.283(0.12)	0.234(0.11)	0.294(0.12)	0.288(0.11)	0.280(0.11)
RF	servo	0.816(0.12)	0.813(0.11)	0.834(0.13)	0.814(0.14)	0.822(0.14)	0.760(0.14)	0.779(0.12)
	a6	0.600(0.15)	0.664(0.17)	0.616(0.15)	0.641(0.16)	0.645(0.17)	0.625(0.15)	0.643(0.16)
	Abalone	0.748(0.02)	0.808(0.02)	0.764(0.02)	0.776(0.02)	0.802(0.02)	0.803(0.02)	0.803(0.02)
	machineCpu	0.853(0.07)	0.856(0.07)	0.850(0.08)	0.843(0.07)	0.855(0.07)	0.855(0.06)	0.857(0.07)
	a3	0.606(0.11)	0.693(0.12)	0.638(0.13)	0.657(0.14)	0.691(0.11)	0.671(0.10)	0.659(0.12)
	a4	0.662(0.11)	0.729(0.14)	0.674(0.11)	0.699(0.13)	0.703(0.11)	0.720(0.11)	0.711(0.10)
	a1	0.772(0.10)	0.834(0.09)	0.786(0.10)	0.801(0.10)	0.809(0.10)	0.821(0.10)	0.815(0.10)

Continued on next page

Table A.2 – continued from previous page

Learner	Data sets	None	RU	RO	WERCS	SMOTER	GN	SMOGN
	a7	0.450(0.18)	0.504(0.23)	0.497(0.22)	0.539(0.23)	0.535(0.21)	0.476(0.23)	0.489(0.21)
	boston	0.920(0.05)	0.930(0.04)	0.928(0.04)	0.930(0.04)	0.928(0.04)	0.933(0.04)	0.930(0.04)
	a2	0.615(0.24)	0.711(0.27)	0.631(0.24)	0.651(0.25)	0.675(0.25)	0.686(0.27)	0.673(0.26)
	a5	0.604(0.24)	0.692(0.26)	0.632(0.26)	0.645(0.26)	0.660(0.26)	0.677(0.27)	0.673(0.27)
	fuelCons	0.922(0.01)	0.931(0.02)	0.940(0.01)	0.941(0.01)	0.942(0.01)	0.940(0.01)	0.941(0.01)
	availPwr	0.973(0.01)	0.972(0.01)	0.984(0.01)	0.984(0.01)	0.977(0.01)	0.982(0.01)	0.978(0.01)
RF	cpuSm	0.566(0.05)	0.584(0.06)	0.579(0.05)	0.570(0.05)	0.581(0.05)	0.581(0.05)	0.579(0.05)
	maxTorque	0.974(0.01)	0.975(0.01)	0.983(0.01)	0.984(0.01)	0.980(0.01)	0.982(0.01)	0.981(0.01)
	bank8FM	0.955(0.01)	0.964(0.01)	0.956(0.01)	0.955(0.01)	0.965(0.00)	0.961(0.01)	0.964(0.00)
	dAiler	0.770(0.02)	0.861(0.02)	0.776(0.02)	0.792(0.02)	0.843(0.02)	0.830(0.02)	0.842(0.02)
	concreteStrength	0.942(0.02)	0.951(0.02)	0.952(0.03)	0.955(0.03)	0.952(0.02)	0.949(0.02)	0.952(0.02)
	acceleration	0.948(0.02)	0.959(0.01)	0.963(0.01)	0.965(0.01)	0.965(0.01)	0.963(0.01)	0.966(0.01)
	airfoild	0.287(0.14)	0.298(0.10)	0.274(0.10)	0.271(0.10)	0.278(0.11)	0.276(0.12)	0.306(0.10)
	servo	0.497(0.14)	0.744(0.12)	0.800(0.12)	0.807(0.11)	0.778(0.11)	0.739(0.13)	0.744(0.12)
	a6	0.550(0.10)	0.640(0.14)	0.639(0.13)	0.653(0.12)	0.634(0.12)	0.638(0.12)	0.638(0.13)
	Abalone	0.717(0.02)	0.794(0.02)	0.795(0.02)	0.788(0.02)	0.809(0.02)	0.807(0.02)	0.809(0.02)
	machineCpu	0.839(0.08)	0.851(0.07)	0.857(0.07)	0.852(0.07)	0.849(0.07)	0.855(0.07)	0.858(0.07)
	a3	0.536(0.07)	0.671(0.12)	0.658(0.10)	0.659(0.11)	0.662(0.12)	0.657(0.11)	0.644(0.11)
	a4	0.597(0.10)	0.696(0.13)	0.685(0.12)	0.687(0.13)	0.682(0.13)	0.703(0.11)	0.688(0.12)
	a1	0.679(0.06)	0.815(0.09)	0.831(0.08)	0.821(0.08)	0.822(0.09)	0.820(0.08)	0.814(0.08)
	a7	0.434(0.14)	0.423(0.22)	0.446(0.20)	0.464(0.21)	0.452(0.21)	0.467(0.21)	0.426(0.21)
SVM	boston	0.900(0.04)	0.923(0.04)	0.936(0.03)	0.936(0.03)	0.930(0.04)	0.933(0.03)	0.932(0.04)
	a2	0.591(0.24)	0.655(0.26)	0.654(0.25)	0.633(0.24)	0.665(0.26)	0.645(0.24)	0.645(0.25)
	a5	0.558(0.21)	0.674(0.27)	0.678(0.27)	0.677(0.27)	0.665(0.26)	0.676(0.27)	0.677(0.27)
	fuelCons	0.921(0.02)	0.927(0.02)	0.928(0.02)	0.926(0.02)	0.927(0.02)	0.928(0.02)	0.929(0.02)
	availPwr	0.950(0.01)	0.955(0.01)	0.960(0.01)	0.960(0.01)	0.957(0.01)	0.946(0.01)	0.954(0.01)
	cpuSm	0.241(0.03)	0.238(0.04)	0.270(0.04)	0.276(0.04)	0.262(0.04)	0.275(0.03)	0.260(0.04)
	maxTorque	0.979(0.01)	0.980(0.00)	0.980(0.00)	0.983(0.01)	0.980(0.01)	0.981(0.01)	0.979(0.01)
	bank8FM	0.955(0.01)	0.955(0.01)	0.959(0.01)	0.961(0.01)	0.958(0.01)	0.958(0.01)	0.958(0.01)
	dAiler	0.740(0.02)	0.848(0.02)	0.854(0.02)	0.820(0.02)	0.864(0.01)	0.862(0.01)	0.864(0.01)

Continued on next page

Table A.2 – continued from previous page

Learner	Data sets	None	RU	RO	WERCS	SMOTER	GN	SMOGN
SVM	concreteStrength	0.899(0.03)	0.943(0.02)	0.953(0.02)	0.947(0.02)	0.953(0.02)	0.953(0.02)	0.953(0.02)
	acceleration	0.907(0.02)	0.949(0.02)	0.946(0.01)	0.938(0.02)	0.946(0.02)	0.942(0.02)	0.944(0.02)
	airfoild	0.237(0.09)	0.294(0.12)	0.331(0.12)	0.297(0.12)	0.318(0.11)	0.309(0.11)	0.305(0.11)
NNET	servo	0.746(0.13)	0.708(0.17)	0.782(0.15)	0.751(0.15)	0.738(0.17)	0.670(0.17)	0.656(0.17)
	a6	0.562(0.12)	0.637(0.16)	0.654(0.15)	0.649(0.16)	0.641(0.16)	0.643(0.16)	0.652(0.15)
	Abalone	0.736(0.03)	0.798(0.04)	0.799(0.05)	0.775(0.05)	0.808(0.05)	0.820(0.02)	0.815(0.03)
	machineCpu	0.582(0.11)	0.689(0.14)	0.690(0.13)	0.707(0.12)	0.700(0.14)	0.701(0.13)	0.693(0.14)
	a3	0.559(0.09)	0.656(0.13)	0.653(0.10)	0.666(0.12)	0.648(0.11)	0.652(0.11)	0.661(0.12)
	a4	0.635(0.12)	0.702(0.13)	0.705(0.14)	0.715(0.12)	0.715(0.12)	0.699(0.12)	0.699(0.12)
	a1	0.694(0.10)	0.779(0.09)	0.787(0.09)	0.784(0.09)	0.787(0.08)	0.773(0.09)	0.774(0.09)
	a7	0.468(0.21)	0.472(0.20)	0.487(0.22)	0.499(0.22)	0.487(0.20)	0.474(0.20)	0.511(0.21)
	boston	0.748(0.07)	0.821(0.05)	0.819(0.05)	0.830(0.05)	0.810(0.05)	0.809(0.05)	0.807(0.05)
	a2	0.574(0.22)	0.675(0.26)	0.672(0.26)	0.676(0.26)	0.671(0.26)	0.681(0.26)	0.680(0.26)
	a5	0.540(0.21)	0.664(0.26)	0.663(0.26)	0.653(0.26)	0.659(0.26)	0.660(0.26)	0.658(0.26)
	fuelCons	0.518(0.04)	0.626(0.05)	0.584(0.07)	0.575(0.05)	0.609(0.08)	0.600(0.07)	0.613(0.08)
	availPwr	0.637(0.05)	0.776(0.05)	0.773(0.05)	0.743(0.04)	0.755(0.05)	0.756(0.05)	0.754(0.04)
	cpuSm	0.241(0.06)	0.279(0.13)	0.250(0.12)	0.272(0.16)	0.257(0.13)	0.227(0.14)	0.301(0.17)
	maxTorque	0.667(0.04)	0.809(0.04)	0.802(0.04)	0.787(0.04)	0.797(0.04)	0.795(0.04)	0.799(0.04)
	bank8FM	0.962(0.01)	0.956(0.02)	0.963(0.01)	0.958(0.02)	0.965(0.01)	0.966(0.01)	0.964(0.01)
	dAiler	0.607(0.02)	0.653(0.03)	0.692(0.05)	0.649(0.03)	0.688(0.03)	0.682(0.05)	0.694(0.03)
	concreteStrength	0.700(0.03)	0.856(0.04)	0.835(0.04)	0.822(0.03)	0.832(0.04)	0.840(0.04)	0.835(0.03)
	acceleration	0.699(0.04)	0.834(0.03)	0.835(0.03)	0.809(0.03)	0.831(0.03)	0.839(0.04)	0.842(0.04)
	airfoild	0.341(0.04)	0.241(0.05)	0.245(0.03)	0.256(0.03)	0.249(0.03)	0.244(0.03)	0.251(0.04)

Table A.3: Evaluation results concerning the $prec^{\phi}$ metric (average and standard deviation) for 20 data sets and all tested variants of 4 learners using unbiased pre-processing strategies.

Learner	Data sets	None	RU	RO	WERCS	SMOTER	GN	SMOGN
MARS	servo	0.614(0.10)	0.585(0.13)	0.605(0.11)	0.595(0.11)	0.611(0.12)	0.557(0.15)	0.581(0.14)
	a6	0.475(0.20)	0.485(0.06)	0.491(0.07)	0.468(0.06)	0.491(0.07)	0.487(0.07)	0.486(0.06)
	Abalone	0.731(0.03)	0.698(0.02)	0.706(0.02)	0.708(0.02)	0.679(0.01)	0.677(0.01)	0.676(0.01)
	machineCpu	0.794(0.11)	0.748(0.09)	0.752(0.13)	0.754(0.11)	0.759(0.10)	0.769(0.09)	0.765(0.09)
	a3	0.531(0.25)	0.463(0.10)	0.480(0.08)	0.462(0.07)	0.467(0.08)	0.463(0.07)	0.476(0.09)
	a4	0.482(0.25)	0.478(0.09)	0.524(0.11)	0.498(0.11)	0.498(0.09)	0.485(0.11)	0.500(0.10)
	a1	0.573(0.28)	0.668(0.11)	0.648(0.11)	0.658(0.10)	0.680(0.11)	0.656(0.10)	0.671(0.11)
	a7	0.307(0.15)	0.338(0.06)	0.348(0.05)	0.351(0.05)	0.346(0.05)	0.343(0.05)	0.350(0.05)
	boston	0.898(0.04)	0.847(0.06)	0.867(0.05)	0.867(0.05)	0.862(0.05)	0.856(0.05)	0.858(0.05)
	a2	0.308(0.41)	0.462(0.13)	0.508(0.11)	0.478(0.13)	0.505(0.08)	0.508(0.09)	0.525(0.08)
	a5	0.156(0.22)	0.466(0.12)	0.522(0.08)	0.535(0.08)	0.520(0.10)	0.504(0.09)	0.518(0.07)
	fuelCons	0.851(0.05)	0.835(0.05)	0.840(0.04)	0.846(0.05)	0.833(0.04)	0.852(0.04)	0.842(0.04)
	availPwr	0.900(0.02)	0.884(0.03)	0.890(0.02)	0.902(0.02)	0.882(0.03)	0.876(0.02)	0.880(0.02)
	cpuSm	0.144(0.03)	0.168(0.02)	0.163(0.03)	0.170(0.04)	0.170(0.02)	0.169(0.02)	0.172(0.02)
	maxTorque	0.954(0.02)	0.974(0.02)	0.955(0.02)	0.968(0.02)	0.953(0.02)	0.959(0.03)	0.960(0.02)
	bank8FM	0.948(0.01)	0.948(0.01)	0.948(0.01)	0.951(0.01)	0.948(0.01)	0.948(0.01)	0.948(0.01)
	dAiler	0.786(0.04)	0.697(0.02)	0.692(0.02)	0.747(0.02)	0.673(0.02)	0.672(0.02)	0.672(0.02)
	concreteStrength	0.895(0.06)	0.847(0.05)	0.867(0.05)	0.889(0.05)	0.868(0.05)	0.860(0.05)	0.873(0.05)
	acceleration	0.892(0.05)	0.843(0.06)	0.855(0.04)	0.892(0.05)	0.865(0.04)	0.856(0.04)	0.864(0.04)
	airfoild	0.118(0.09)	0.215(0.09)	0.201(0.11)	0.180(0.13)	0.218(0.12)	0.211(0.11)	0.205(0.11)
RF	servo	0.750(0.14)	0.691(0.12)	0.757(0.14)	0.735(0.16)	0.730(0.15)	0.614(0.17)	0.641(0.14)
	a6	0.549(0.14)	0.482(0.07)	0.509(0.12)	0.495(0.09)	0.490(0.10)	0.485(0.07)	0.493(0.08)
	Abalone	0.736(0.03)	0.675(0.02)	0.722(0.02)	0.711(0.02)	0.682(0.02)	0.679(0.02)	0.679(0.01)
	machineCpu	0.783(0.09)	0.761(0.10)	0.780(0.11)	0.761(0.11)	0.770(0.09)	0.764(0.09)	0.778(0.10)
	a3	0.459(0.22)	0.460(0.06)	0.568(0.16)	0.496(0.10)	0.511(0.08)	0.488(0.08)	0.496(0.09)
	a4	0.496(0.27)	0.516(0.10)	0.519(0.26)	0.505(0.16)	0.530(0.14)	0.521(0.17)	0.551(0.17)
	a1	0.644(0.32)	0.683(0.07)	0.628(0.24)	0.646(0.17)	0.696(0.09)	0.680(0.12)	0.699(0.12)

Continued on next page

Table A.3 – continued from previous page

Learner	Data sets	None	RU	RO	WERCS	SMOTER	GN	SMOGN
	a7	0.314(0.18)	0.365(0.04)	0.338(0.14)	0.381(0.11)	0.379(0.06)	0.368(0.07)	0.355(0.10)
	boston	0.909(0.04)	0.873(0.05)	0.911(0.04)	0.913(0.04)	0.884(0.04)	0.880(0.05)	0.893(0.04)
	a2	0.291(0.38)	0.551(0.09)	0.443(0.29)	0.481(0.28)	0.561(0.14)	0.604(0.17)	0.633(0.19)
	a5	0.244(0.32)	0.518(0.05)	0.497(0.24)	0.532(0.19)	0.554(0.11)	0.551(0.11)	0.569(0.10)
	fuelCons	0.928(0.02)	0.864(0.03)	0.939(0.02)	0.936(0.03)	0.904(0.03)	0.910(0.04)	0.912(0.03)
	availPwr	0.962(0.02)	0.936(0.03)	0.977(0.02)	0.977(0.02)	0.962(0.02)	0.969(0.02)	0.966(0.01)
RF	cpuSm	0.525(0.05)	0.455(0.06)	0.529(0.06)	0.528(0.06)	0.486(0.06)	0.490(0.05)	0.486(0.06)
	maxTorque	0.969(0.01)	0.928(0.03)	0.977(0.01)	0.981(0.01)	0.961(0.02)	0.965(0.02)	0.965(0.02)
	bank8FM	0.947(0.01)	0.926(0.01)	0.944(0.01)	0.945(0.01)	0.936(0.01)	0.936(0.01)	0.936(0.01)
	dAiler	0.752(0.04)	0.684(0.02)	0.711(0.04)	0.724(0.04)	0.688(0.02)	0.701(0.03)	0.693(0.02)
	concreteStrength	0.933(0.22)	0.875(0.04)	0.971(0.03)	0.925(0.22)	0.944(0.04)	0.942(0.04)	0.957(0.03)
	acceleration	0.933(0.02)	0.865(0.03)	0.947(0.02)	0.951(0.02)	0.925(0.03)	0.937(0.03)	0.928(0.03)
	airfoild	0.244(0.18)	0.214(0.09)	0.207(0.11)	0.215(0.10)	0.201(0.10)	0.193(0.11)	0.238(0.10)
	servo	0.369(0.11)	0.538(0.07)	0.592(0.08)	0.607(0.08)	0.572(0.07)	0.535(0.10)	0.544(0.09)
	a6	0.258(0.23)	0.492(0.08)	0.500(0.09)	0.495(0.09)	0.494(0.08)	0.516(0.08)	0.511(0.09)
	Abalone	0.767(0.04)	0.708(0.02)	0.714(0.02)	0.723(0.02)	0.688(0.02)	0.696(0.02)	0.690(0.01)
	machineCpu	0.770(0.11)	0.763(0.08)	0.763(0.10)	0.775(0.09)	0.758(0.09)	0.769(0.09)	0.772(0.08)
	a3	0.187(0.11)	0.452(0.08)	0.470(0.07)	0.455(0.07)	0.478(0.08)	0.466(0.06)	0.462(0.08)
	a4	0.256(0.26)	0.518(0.09)	0.519(0.10)	0.507(0.08)	0.551(0.12)	0.528(0.12)	0.529(0.12)
	a1	0.103(0.16)	0.620(0.08)	0.651(0.09)	0.654(0.07)	0.660(0.11)	0.661(0.12)	0.651(0.11)
	a7	0.141(0.13)	0.349(0.05)	0.345(0.05)	0.358(0.06)	0.347(0.06)	0.369(0.08)	0.354(0.07)
SVM	boston	0.895(0.04)	0.855(0.04)	0.884(0.03)	0.886(0.04)	0.881(0.04)	0.876(0.04)	0.881(0.04)
	a2	0.282(0.38)	0.492(0.12)	0.519(0.12)	0.514(0.18)	0.501(0.15)	0.522(0.17)	0.529(0.17)
	a5	0.145(0.16)	0.513(0.09)	0.521(0.14)	0.544(0.12)	0.552(0.11)	0.562(0.11)	0.542(0.08)
	fuelCons	0.908(0.04)	0.844(0.04)	0.897(0.04)	0.899(0.04)	0.869(0.03)	0.874(0.04)	0.876(0.04)
	availPwr	0.934(0.02)	0.906(0.04)	0.936(0.02)	0.944(0.02)	0.930(0.01)	0.911(0.03)	0.925(0.02)
	cpuSm	0.163(0.03)	0.178(0.03)	0.187(0.03)	0.195(0.03)	0.180(0.03)	0.193(0.03)	0.179(0.03)
	maxTorque	0.973(0.01)	0.942(0.03)	0.972(0.01)	0.976(0.01)	0.969(0.01)	0.969(0.01)	0.968(0.01)
	bank8FM	0.949(0.01)	0.952(0.01)	0.949(0.01)	0.953(0.01)	0.951(0.01)	0.951(0.01)	0.951(0.01)
	dAiler	0.781(0.05)	0.711(0.02)	0.698(0.02)	0.749(0.03)	0.677(0.02)	0.681(0.02)	0.677(0.02)

Continued on next page

Table A.3 – continued from previous page

Learner	Data sets	None	RU	RO	WERCS	SMOTER	GN	SMOGN
SVM	concreteStrength	0.855(0.15)	0.827(0.07)	0.875(0.05)	0.897(0.06)	0.877(0.05)	0.872(0.05)	0.870(0.05)
	acceleration	0.876(0.08)	0.864(0.06)	0.865(0.04)	0.907(0.03)	0.869(0.04)	0.891(0.03)	0.874(0.04)
	airfoild	0.166(0.11)	0.222(0.12)	0.265(0.15)	0.233(0.12)	0.247(0.11)	0.238(0.12)	0.239(0.12)
NNET	servo	0.644(0.14)	0.587(0.16)	0.708(0.17)	0.659(0.19)	0.648(0.18)	0.521(0.16)	0.525(0.16)
	a6	0.175(0.19)	0.470(0.07)	0.465(0.07)	0.465(0.06)	0.473(0.06)	0.461(0.09)	0.471(0.07)
	Abalone	0.707(0.06)	0.656(0.08)	0.663(0.10)	0.633(0.10)	0.650(0.08)	0.671(0.02)	0.660(0.05)
	machineCpu	0.059(0.07)	0.454(0.07)	0.446(0.08)	0.457(0.08)	0.484(0.07)	0.466(0.07)	0.461(0.07)
	a3	0.203(0.22)	0.434(0.05)	0.434(0.06)	0.430(0.05)	0.453(0.06)	0.439(0.06)	0.456(0.05)
	a4	0.407(0.25)	0.427(0.13)	0.406(0.17)	0.442(0.15)	0.479(0.13)	0.434(0.16)	0.438(0.12)
	a1	0.252(0.27)	0.400(0.24)	0.455(0.22)	0.327(0.26)	0.386(0.25)	0.421(0.27)	0.380(0.27)
	a7	0.282(0.22)	0.326(0.05)	0.351(0.07)	0.346(0.07)	0.334(0.06)	0.338(0.06)	0.352(0.06)
	boston	0.308(0.28)	0.324(0.20)	0.311(0.26)	0.346(0.27)	0.276(0.22)	0.280(0.23)	0.253(0.25)
	a2	0.048(0.14)	0.325(0.15)	0.316(0.21)	0.344(0.22)	0.369(0.19)	0.421(0.19)	0.365(0.21)
	a5	0.011(0.04)	0.363(0.15)	0.380(0.18)	0.322(0.21)	0.362(0.17)	0.416(0.12)	0.353(0.18)
	fuelCons	0.065(0.15)	0.368(0.22)	0.255(0.20)	0.171(0.22)	0.290(0.24)	0.269(0.18)	0.280(0.22)
	availPwr	0.094(0.12)	0.166(0.20)	0.173(0.26)	0.035(0.09)	0.056(0.13)	0.076(0.16)	0.047(0.10)
	cpuSm	0.094(0.10)	0.317(0.24)	0.262(0.21)	0.340(0.33)	0.316(0.25)	0.218(0.22)	0.370(0.30)
	maxTorque	0.093(0.18)	0.397(0.21)	0.261(0.23)	0.273(0.25)	0.237(0.16)	0.245(0.27)	0.370(0.24)
	bank8FM	0.954(0.01)	0.890(0.10)	0.925(0.05)	0.925(0.08)	0.938(0.03)	0.937(0.03)	0.933(0.06)
	dAiler	0.351(0.09)	0.301(0.04)	0.348(0.12)	0.322(0.05)	0.334(0.06)	0.332(0.11)	0.364(0.09)
	concreteStrength	0.000(0.00)	0.258(0.20)	0.115(0.22)	0.053(0.11)	0.115(0.18)	0.164(0.17)	0.150(0.16)
	acceleration	0.116(0.29)	0.268(0.20)	0.299(0.28)	0.174(0.24)	0.235(0.23)	0.298(0.24)	0.349(0.26)
	airfoild	0.020(0.03)	0.078(0.07)	0.032(0.05)	0.014(0.03)	0.029(0.04)	0.038(0.04)	0.046(0.06)

Table A.4: Evaluation results concerning the rec^{ϕ} metric (average and standard deviation) for 20 data sets and all tested variants of 4 learners using unbiased pre-processing strategies.

Learner	Data sets	None	RU	RO	WERCS	SMOTER	GN	SMOGN
MARS	servo	0.614(0.10)	0.585(0.13)	0.605(0.11)	0.595(0.11)	0.611(0.12)	0.557(0.15)	0.581(0.14)
	a6	0.475(0.20)	0.485(0.06)	0.491(0.07)	0.468(0.06)	0.491(0.07)	0.487(0.07)	0.486(0.06)
	Abalone	0.731(0.03)	0.698(0.02)	0.706(0.02)	0.708(0.02)	0.679(0.01)	0.677(0.01)	0.676(0.01)
	machineCpu	0.794(0.11)	0.748(0.09)	0.752(0.13)	0.754(0.11)	0.759(0.10)	0.769(0.09)	0.765(0.09)
	a3	0.531(0.25)	0.463(0.10)	0.480(0.08)	0.462(0.07)	0.467(0.08)	0.463(0.07)	0.476(0.09)
	a4	0.482(0.25)	0.478(0.09)	0.524(0.11)	0.498(0.11)	0.498(0.09)	0.485(0.11)	0.500(0.10)
	a1	0.573(0.28)	0.668(0.11)	0.648(0.11)	0.658(0.10)	0.680(0.11)	0.656(0.10)	0.671(0.11)
	a7	0.307(0.15)	0.338(0.06)	0.348(0.05)	0.351(0.05)	0.346(0.05)	0.343(0.05)	0.350(0.05)
	boston	0.898(0.04)	0.847(0.06)	0.867(0.05)	0.867(0.05)	0.862(0.05)	0.856(0.05)	0.858(0.05)
	a2	0.308(0.41)	0.462(0.13)	0.508(0.11)	0.478(0.13)	0.505(0.08)	0.508(0.09)	0.525(0.08)
	a5	0.156(0.22)	0.466(0.12)	0.522(0.08)	0.535(0.08)	0.520(0.10)	0.504(0.09)	0.518(0.07)
	fuelCons	0.851(0.05)	0.835(0.05)	0.840(0.04)	0.846(0.05)	0.833(0.04)	0.852(0.04)	0.842(0.04)
	availPwr	0.900(0.02)	0.884(0.03)	0.890(0.02)	0.902(0.02)	0.882(0.03)	0.876(0.02)	0.880(0.02)
	cpuSm	0.144(0.03)	0.168(0.02)	0.163(0.03)	0.170(0.04)	0.170(0.02)	0.169(0.02)	0.172(0.02)
	maxTorque	0.954(0.02)	0.974(0.02)	0.955(0.02)	0.968(0.02)	0.953(0.02)	0.959(0.03)	0.960(0.02)
	bank8FM	0.948(0.01)	0.948(0.01)	0.948(0.01)	0.951(0.01)	0.948(0.01)	0.948(0.01)	0.948(0.01)
	dAiler	0.786(0.04)	0.697(0.02)	0.692(0.02)	0.747(0.02)	0.673(0.02)	0.672(0.02)	0.672(0.02)
	concreteStrength	0.895(0.06)	0.847(0.05)	0.867(0.05)	0.889(0.05)	0.868(0.05)	0.860(0.05)	0.873(0.05)
	acceleration	0.892(0.05)	0.843(0.06)	0.855(0.04)	0.892(0.05)	0.865(0.04)	0.856(0.04)	0.864(0.04)
	airfoild	0.118(0.09)	0.215(0.09)	0.201(0.11)	0.180(0.13)	0.218(0.12)	0.211(0.11)	0.205(0.11)
RF	servo	0.750(0.14)	0.691(0.12)	0.757(0.14)	0.735(0.16)	0.730(0.15)	0.614(0.17)	0.641(0.14)
	a6	0.549(0.14)	0.482(0.07)	0.509(0.12)	0.495(0.09)	0.490(0.10)	0.485(0.07)	0.493(0.08)
	Abalone	0.736(0.03)	0.675(0.02)	0.722(0.02)	0.711(0.02)	0.682(0.02)	0.679(0.02)	0.679(0.01)
	machineCpu	0.783(0.09)	0.761(0.10)	0.780(0.11)	0.761(0.11)	0.770(0.09)	0.764(0.09)	0.778(0.10)
	a3	0.459(0.22)	0.460(0.06)	0.568(0.16)	0.496(0.10)	0.511(0.08)	0.488(0.08)	0.496(0.09)
	a4	0.496(0.27)	0.516(0.10)	0.519(0.26)	0.505(0.16)	0.530(0.14)	0.521(0.17)	0.551(0.17)
	a1	0.644(0.32)	0.683(0.07)	0.628(0.24)	0.646(0.17)	0.696(0.09)	0.680(0.12)	0.699(0.12)

Continued on next page

Table A.4 – continued from previous page

Learner	Data sets	None	RU	RO	WERCS	SMOTER	GN	SMOGN
	a7	0.314(0.18)	0.365(0.04)	0.338(0.14)	0.381(0.11)	0.379(0.06)	0.368(0.07)	0.355(0.10)
	boston	0.909(0.04)	0.873(0.05)	0.911(0.04)	0.913(0.04)	0.884(0.04)	0.880(0.05)	0.893(0.04)
	a2	0.291(0.38)	0.551(0.09)	0.443(0.29)	0.481(0.28)	0.561(0.14)	0.604(0.17)	0.633(0.19)
	a5	0.244(0.32)	0.518(0.05)	0.497(0.24)	0.532(0.19)	0.554(0.11)	0.551(0.11)	0.569(0.10)
	fuelCons	0.928(0.02)	0.864(0.03)	0.939(0.02)	0.936(0.03)	0.904(0.03)	0.910(0.04)	0.912(0.03)
	availPwr	0.962(0.02)	0.936(0.03)	0.977(0.02)	0.977(0.02)	0.962(0.02)	0.969(0.02)	0.966(0.01)
RF	cpuSm	0.525(0.05)	0.455(0.06)	0.529(0.06)	0.528(0.06)	0.486(0.06)	0.490(0.05)	0.486(0.06)
	maxTorque	0.969(0.01)	0.928(0.03)	0.977(0.01)	0.981(0.01)	0.961(0.02)	0.965(0.02)	0.965(0.02)
	bank8FM	0.947(0.01)	0.926(0.01)	0.944(0.01)	0.945(0.01)	0.936(0.01)	0.936(0.01)	0.936(0.01)
	dAiler	0.752(0.04)	0.684(0.02)	0.711(0.04)	0.724(0.04)	0.688(0.02)	0.701(0.03)	0.693(0.02)
	concreteStrength	0.933(0.22)	0.875(0.04)	0.971(0.03)	0.925(0.22)	0.944(0.04)	0.942(0.04)	0.957(0.03)
	acceleration	0.933(0.02)	0.865(0.03)	0.947(0.02)	0.951(0.02)	0.925(0.03)	0.937(0.03)	0.928(0.03)
	airfoild	0.244(0.18)	0.214(0.09)	0.207(0.11)	0.215(0.10)	0.201(0.10)	0.193(0.11)	0.238(0.10)
	servo	0.369(0.11)	0.538(0.07)	0.592(0.08)	0.607(0.08)	0.572(0.07)	0.535(0.10)	0.544(0.09)
	a6	0.258(0.23)	0.492(0.08)	0.500(0.09)	0.495(0.09)	0.494(0.08)	0.516(0.08)	0.511(0.09)
	Abalone	0.767(0.04)	0.708(0.02)	0.714(0.02)	0.723(0.02)	0.688(0.02)	0.696(0.02)	0.690(0.01)
	machineCpu	0.770(0.11)	0.763(0.08)	0.763(0.10)	0.775(0.09)	0.758(0.09)	0.769(0.09)	0.772(0.08)
	a3	0.187(0.11)	0.452(0.08)	0.470(0.07)	0.455(0.07)	0.478(0.08)	0.466(0.06)	0.462(0.08)
	a4	0.256(0.26)	0.518(0.09)	0.519(0.10)	0.507(0.08)	0.551(0.12)	0.528(0.12)	0.529(0.12)
	a1	0.103(0.16)	0.620(0.08)	0.651(0.09)	0.654(0.07)	0.660(0.11)	0.661(0.12)	0.651(0.11)
	a7	0.141(0.13)	0.349(0.05)	0.345(0.05)	0.358(0.06)	0.347(0.06)	0.369(0.08)	0.354(0.07)
SVM	boston	0.895(0.04)	0.855(0.04)	0.884(0.03)	0.886(0.04)	0.881(0.04)	0.876(0.04)	0.881(0.04)
	a2	0.282(0.38)	0.492(0.12)	0.519(0.12)	0.514(0.18)	0.501(0.15)	0.522(0.17)	0.529(0.17)
	a5	0.145(0.16)	0.513(0.09)	0.521(0.14)	0.544(0.12)	0.552(0.11)	0.562(0.11)	0.542(0.08)
	fuelCons	0.908(0.04)	0.844(0.04)	0.897(0.04)	0.899(0.04)	0.869(0.03)	0.874(0.04)	0.876(0.04)
	availPwr	0.934(0.02)	0.906(0.04)	0.936(0.02)	0.944(0.02)	0.930(0.01)	0.911(0.03)	0.925(0.02)
	cpuSm	0.163(0.03)	0.178(0.03)	0.187(0.03)	0.195(0.03)	0.180(0.03)	0.193(0.03)	0.179(0.03)
	maxTorque	0.973(0.01)	0.942(0.03)	0.972(0.01)	0.976(0.01)	0.969(0.01)	0.969(0.01)	0.968(0.01)
	bank8FM	0.949(0.01)	0.952(0.01)	0.949(0.01)	0.953(0.01)	0.951(0.01)	0.951(0.01)	0.951(0.01)
	dAiler	0.781(0.05)	0.711(0.02)	0.698(0.02)	0.749(0.03)	0.677(0.02)	0.681(0.02)	0.677(0.02)

Continued on next page

Table A.4 – continued from previous page

Learner	Data sets	None	RU	RO	WERCS	SMOTER	GN	SMOGN
SVM	concreteStrength	0.855(0.15)	0.827(0.07)	0.875(0.05)	0.897(0.06)	0.877(0.05)	0.872(0.05)	0.870(0.05)
	acceleration	0.876(0.08)	0.864(0.06)	0.865(0.04)	0.907(0.03)	0.869(0.04)	0.891(0.03)	0.874(0.04)
	airfoild	0.166(0.11)	0.222(0.12)	0.265(0.15)	0.233(0.12)	0.247(0.11)	0.238(0.12)	0.239(0.12)
NNET	servo	0.644(0.14)	0.587(0.16)	0.708(0.17)	0.659(0.19)	0.648(0.18)	0.521(0.16)	0.525(0.16)
	a6	0.175(0.19)	0.470(0.07)	0.465(0.07)	0.465(0.06)	0.473(0.06)	0.461(0.09)	0.471(0.07)
	Abalone	0.707(0.06)	0.656(0.08)	0.663(0.10)	0.633(0.10)	0.650(0.08)	0.671(0.02)	0.660(0.05)
	machineCpu	0.059(0.07)	0.454(0.07)	0.446(0.08)	0.457(0.08)	0.484(0.07)	0.466(0.07)	0.461(0.07)
	a3	0.203(0.22)	0.434(0.05)	0.434(0.06)	0.430(0.05)	0.453(0.06)	0.439(0.06)	0.456(0.05)
	a4	0.407(0.25)	0.427(0.13)	0.406(0.17)	0.442(0.15)	0.479(0.13)	0.434(0.16)	0.438(0.12)
	a1	0.252(0.27)	0.400(0.24)	0.455(0.22)	0.327(0.26)	0.386(0.25)	0.421(0.27)	0.380(0.27)
	a7	0.282(0.22)	0.326(0.05)	0.351(0.07)	0.346(0.07)	0.334(0.06)	0.338(0.06)	0.352(0.06)
	boston	0.308(0.28)	0.324(0.20)	0.311(0.26)	0.346(0.27)	0.276(0.22)	0.280(0.23)	0.253(0.25)
	a2	0.048(0.14)	0.325(0.15)	0.316(0.21)	0.344(0.22)	0.369(0.19)	0.421(0.19)	0.365(0.21)
	a5	0.011(0.04)	0.363(0.15)	0.380(0.18)	0.322(0.21)	0.362(0.17)	0.416(0.12)	0.353(0.18)
	fuelCons	0.065(0.15)	0.368(0.22)	0.255(0.20)	0.171(0.22)	0.290(0.24)	0.269(0.18)	0.280(0.22)
	availPwr	0.094(0.12)	0.166(0.20)	0.173(0.26)	0.035(0.09)	0.056(0.13)	0.076(0.16)	0.047(0.10)
	cpuSm	0.094(0.10)	0.317(0.24)	0.262(0.21)	0.340(0.33)	0.316(0.25)	0.218(0.22)	0.370(0.30)
	maxTorque	0.093(0.18)	0.397(0.21)	0.261(0.23)	0.273(0.25)	0.237(0.16)	0.245(0.27)	0.370(0.24)
	bank8FM	0.954(0.01)	0.890(0.10)	0.925(0.05)	0.925(0.08)	0.938(0.03)	0.937(0.03)	0.933(0.06)
	dAiler	0.351(0.09)	0.301(0.04)	0.348(0.12)	0.322(0.05)	0.334(0.06)	0.332(0.11)	0.364(0.09)
	concreteStrength	0.000(0.00)	0.258(0.20)	0.115(0.22)	0.053(0.11)	0.115(0.18)	0.164(0.17)	0.150(0.16)
	acceleration	0.116(0.29)	0.268(0.20)	0.299(0.28)	0.174(0.24)	0.235(0.23)	0.298(0.24)	0.349(0.26)
	airfoild	0.020(0.03)	0.078(0.07)	0.032(0.05)	0.014(0.03)	0.029(0.04)	0.038(0.04)	0.046(0.06)

Table A.5: Evaluation results concerning the $spec^{\phi}$ metric (average and standard deviation) for 20 data sets and all tested variants of 4 learners using unbiased pre-processing strategies.

Learner	Data sets	None	RU	RO	WERCS	SMOTER	GN	SMOGN
MARS	servo	0.792(0.09)	0.788(0.09)	0.803(0.08)	0.790(0.09)	0.805(0.09)	0.781(0.10)	0.795(0.09)
	a6	0.690(0.09)	0.696(0.10)	0.700(0.09)	0.695(0.09)	0.710(0.09)	0.697(0.10)	0.700(0.10)
	Abalone	0.784(0.01)	0.828(0.01)	0.827(0.01)	0.825(0.01)	0.832(0.01)	0.830(0.01)	0.830(0.01)
	machineCpu	0.887(0.05)	0.888(0.05)	0.887(0.05)	0.885(0.04)	0.887(0.05)	0.896(0.04)	0.891(0.05)
	a3	0.695(0.10)	0.716(0.08)	0.723(0.10)	0.720(0.09)	0.716(0.08)	0.723(0.08)	0.721(0.09)
	a4	0.720(0.08)	0.756(0.10)	0.758(0.11)	0.747(0.10)	0.746(0.09)	0.732(0.10)	0.754(0.09)
	a1	0.816(0.07)	0.856(0.07)	0.855(0.06)	0.847(0.06)	0.851(0.07)	0.850(0.06)	0.865(0.07)
	a7	0.544(0.18)	0.547(0.18)	0.530(0.18)	0.530(0.17)	0.540(0.19)	0.517(0.19)	0.517(0.22)
	boston	0.937(0.03)	0.941(0.03)	0.945(0.03)	0.944(0.03)	0.943(0.03)	0.943(0.03)	0.943(0.03)
	a2	0.691(0.25)	0.707(0.25)	0.696(0.26)	0.700(0.25)	0.702(0.25)	0.712(0.26)	0.709(0.26)
	a5	0.664(0.24)	0.701(0.25)	0.707(0.25)	0.706(0.26)	0.706(0.26)	0.685(0.26)	0.702(0.25)
	fuelCons	0.900(0.01)	0.923(0.01)	0.925(0.01)	0.917(0.01)	0.924(0.01)	0.930(0.01)	0.926(0.01)
	availPwr	0.946(0.01)	0.952(0.01)	0.952(0.01)	0.952(0.01)	0.951(0.01)	0.950(0.01)	0.949(0.01)
	cpuSm	0.345(0.04)	0.369(0.03)	0.361(0.04)	0.371(0.04)	0.373(0.03)	0.370(0.03)	0.375(0.03)
	maxTorque	0.973(0.01)	0.991(0.00)	0.978(0.01)	0.983(0.01)	0.980(0.01)	0.981(0.01)	0.982(0.01)
	bank8FM	0.961(0.00)	0.967(0.00)	0.967(0.00)	0.967(0.00)	0.966(0.00)	0.967(0.00)	0.966(0.00)
	dAiler	0.807(0.01)	0.877(0.01)	0.878(0.01)	0.861(0.01)	0.881(0.01)	0.882(0.01)	0.882(0.01)
	concreteStrength	0.929(0.02)	0.954(0.02)	0.959(0.01)	0.954(0.02)	0.956(0.02)	0.959(0.01)	0.959(0.01)
	acceleration	0.941(0.01)	0.960(0.01)	0.958(0.01)	0.954(0.01)	0.959(0.01)	0.957(0.01)	0.957(0.01)
	airfoild	0.346(0.08)	0.394(0.10)	0.394(0.11)	0.346(0.11)	0.407(0.11)	0.399(0.10)	0.392(0.11)
RF	servo	0.864(0.09)	0.853(0.08)	0.877(0.09)	0.860(0.11)	0.865(0.10)	0.808(0.11)	0.825(0.09)
	a6	0.692(0.12)	0.717(0.12)	0.701(0.12)	0.714(0.12)	0.715(0.13)	0.696(0.11)	0.713(0.12)
	Abalone	0.794(0.01)	0.830(0.01)	0.804(0.01)	0.815(0.01)	0.827(0.01)	0.828(0.01)	0.828(0.01)
	machineCpu	0.893(0.05)	0.893(0.05)	0.891(0.05)	0.886(0.05)	0.894(0.05)	0.894(0.04)	0.895(0.05)
	a3	0.698(0.09)	0.723(0.09)	0.717(0.10)	0.719(0.11)	0.744(0.08)	0.725(0.08)	0.720(0.09)
	a4	0.738(0.08)	0.773(0.10)	0.748(0.08)	0.765(0.09)	0.762(0.08)	0.775(0.08)	0.773(0.08)
	a1	0.828(0.07)	0.864(0.06)	0.836(0.07)	0.847(0.07)	0.850(0.07)	0.858(0.07)	0.855(0.07)

Continued on next page

Table A.5 – continued from previous page

Learner	Data sets	None	RU	RO	WERCS	SMOTER	GN	SMOGN
	a7	0.555(0.18)	0.542(0.19)	0.587(0.20)	0.615(0.20)	0.592(0.19)	0.545(0.22)	0.567(0.20)
	boston	0.942(0.03)	0.948(0.03)	0.947(0.03)	0.949(0.02)	0.947(0.03)	0.950(0.03)	0.948(0.03)
	a2	0.686(0.25)	0.738(0.26)	0.697(0.25)	0.711(0.26)	0.724(0.26)	0.733(0.27)	0.725(0.26)
	a5	0.677(0.25)	0.715(0.26)	0.694(0.26)	0.703(0.26)	0.710(0.26)	0.721(0.26)	0.719(0.26)
	fuelCons	0.937(0.01)	0.935(0.01)	0.949(0.01)	0.951(0.01)	0.947(0.01)	0.945(0.01)	0.947(0.01)
	availPwr	0.981(0.01)	0.977(0.01)	0.989(0.01)	0.989(0.01)	0.983(0.01)	0.986(0.01)	0.984(0.01)
RF	cpuSm	0.651(0.04)	0.660(0.04)	0.661(0.04)	0.653(0.04)	0.661(0.04)	0.661(0.04)	0.659(0.04)
	maxTorque	0.982(0.01)	0.980(0.01)	0.988(0.01)	0.989(0.01)	0.985(0.00)	0.986(0.00)	0.986(0.01)
	bank8FM	0.966(0.00)	0.971(0.00)	0.967(0.00)	0.966(0.00)	0.972(0.00)	0.969(0.00)	0.972(0.00)
	dAiler	0.825(0.01)	0.885(0.01)	0.829(0.01)	0.841(0.01)	0.874(0.01)	0.865(0.01)	0.874(0.01)
	concreteStrength	0.957(0.02)	0.959(0.02)	0.964(0.02)	0.966(0.02)	0.963(0.02)	0.961(0.02)	0.963(0.02)
	acceleration	0.962(0.01)	0.967(0.01)	0.972(0.01)	0.974(0.01)	0.973(0.01)	0.971(0.01)	0.973(0.01)
	airfoild	0.400(0.14)	0.409(0.10)	0.392(0.10)	0.389(0.10)	0.391(0.11)	0.390(0.12)	0.422(0.10)
	servo	0.601(0.13)	0.774(0.09)	0.822(0.08)	0.831(0.08)	0.804(0.08)	0.770(0.09)	0.776(0.09)
	a6	0.657(0.08)	0.709(0.11)	0.712(0.10)	0.720(0.09)	0.711(0.09)	0.714(0.09)	0.716(0.09)
	Abalone	0.773(0.01)	0.825(0.01)	0.827(0.01)	0.823(0.01)	0.833(0.01)	0.832(0.01)	0.833(0.01)
	machineCpu	0.883(0.05)	0.890(0.05)	0.895(0.05)	0.891(0.05)	0.889(0.05)	0.893(0.05)	0.896(0.04)
	a3	0.644(0.06)	0.716(0.09)	0.713(0.08)	0.708(0.08)	0.720(0.09)	0.712(0.08)	0.706(0.09)
	a4	0.688(0.08)	0.749(0.09)	0.745(0.09)	0.744(0.09)	0.746(0.10)	0.760(0.08)	0.749(0.09)
	a1	0.759(0.05)	0.844(0.06)	0.858(0.05)	0.853(0.06)	0.853(0.06)	0.850(0.06)	0.848(0.06)
	a7	0.549(0.15)	0.487(0.21)	0.515(0.20)	0.527(0.21)	0.521(0.20)	0.540(0.20)	0.507(0.20)
SVM	boston	0.928(0.03)	0.942(0.03)	0.952(0.02)	0.952(0.02)	0.948(0.03)	0.950(0.02)	0.950(0.02)
	a2	0.664(0.25)	0.696(0.25)	0.701(0.25)	0.687(0.25)	0.711(0.26)	0.697(0.25)	0.700(0.25)
	a5	0.641(0.23)	0.707(0.26)	0.717(0.26)	0.719(0.26)	0.711(0.26)	0.718(0.26)	0.720(0.26)
	fuelCons	0.935(0.01)	0.933(0.01)	0.938(0.01)	0.939(0.01)	0.936(0.01)	0.937(0.01)	0.938(0.01)
	availPwr	0.964(0.01)	0.966(0.01)	0.970(0.01)	0.971(0.01)	0.968(0.01)	0.961(0.01)	0.967(0.01)
	cpuSm	0.367(0.04)	0.355(0.04)	0.392(0.04)	0.399(0.04)	0.384(0.03)	0.396(0.03)	0.382(0.03)
	maxTorque	0.984(0.00)	0.984(0.00)	0.985(0.00)	0.987(0.00)	0.984(0.00)	0.985(0.01)	0.984(0.00)
	bank8FM	0.965(0.00)	0.966(0.00)	0.968(0.00)	0.970(0.00)	0.968(0.00)	0.968(0.00)	0.968(0.00)
	dAiler	0.800(0.01)	0.877(0.01)	0.881(0.01)	0.860(0.01)	0.886(0.01)	0.885(0.01)	0.886(0.01)

Continued on next page

Table A.5 – continued from previous page

Learner	Data sets	None	RU	RO	WERCS	SMOTER	GN	SMOGN
SVM	concreteStrength	0.925(0.02)	0.953(0.02)	0.962(0.01)	0.959(0.01)	0.962(0.01)	0.961(0.01)	0.962(0.01)
	acceleration	0.933(0.01)	0.961(0.01)	0.959(0.01)	0.955(0.01)	0.959(0.01)	0.957(0.01)	0.958(0.01)
	airfoild	0.354(0.10)	0.403(0.12)	0.440(0.11)	0.408(0.12)	0.429(0.11)	0.420(0.11)	0.417(0.11)
NNET	servo	0.802(0.10)	0.756(0.14)	0.828(0.12)	0.792(0.13)	0.787(0.14)	0.726(0.14)	0.713(0.14)
	a6	0.665(0.10)	0.690(0.12)	0.702(0.11)	0.697(0.12)	0.698(0.12)	0.697(0.12)	0.706(0.11)
	Abalone	0.785(0.02)	0.822(0.03)	0.824(0.03)	0.809(0.04)	0.828(0.03)	0.837(0.01)	0.833(0.02)
	machineCpu	0.685(0.09)	0.711(0.10)	0.707(0.09)	0.719(0.08)	0.730(0.10)	0.723(0.09)	0.718(0.10)
	a3	0.662(0.07)	0.690(0.09)	0.693(0.07)	0.692(0.08)	0.695(0.08)	0.693(0.08)	0.702(0.08)
	a4	0.714(0.09)	0.740(0.10)	0.746(0.10)	0.755(0.08)	0.760(0.08)	0.745(0.09)	0.746(0.09)
	a1	0.767(0.08)	0.815(0.07)	0.824(0.06)	0.816(0.06)	0.824(0.06)	0.814(0.07)	0.816(0.06)
	a7	0.562(0.20)	0.520(0.19)	0.533(0.21)	0.544(0.19)	0.537(0.18)	0.527(0.19)	0.560(0.19)
	boston	0.814(0.05)	0.855(0.04)	0.852(0.04)	0.858(0.04)	0.848(0.04)	0.847(0.04)	0.845(0.04)
	a2	0.654(0.23)	0.703(0.25)	0.706(0.25)	0.710(0.25)	0.705(0.26)	0.714(0.26)	0.714(0.25)
	a5	0.628(0.23)	0.687(0.25)	0.694(0.25)	0.692(0.25)	0.691(0.25)	0.689(0.25)	0.690(0.25)
	fuelCons	0.602(0.04)	0.684(0.04)	0.655(0.05)	0.650(0.04)	0.672(0.06)	0.664(0.05)	0.675(0.06)
	availPwr	0.729(0.04)	0.815(0.04)	0.808(0.04)	0.792(0.03)	0.796(0.04)	0.798(0.04)	0.796(0.03)
	cpuSm	0.359(0.06)	0.364(0.13)	0.338(0.11)	0.356(0.15)	0.343(0.13)	0.319(0.13)	0.383(0.15)
	maxTorque	0.751(0.03)	0.846(0.03)	0.838(0.03)	0.834(0.03)	0.837(0.03)	0.834(0.03)	0.838(0.03)
	bank8FM	0.970(0.00)	0.965(0.02)	0.971(0.01)	0.968(0.01)	0.972(0.01)	0.973(0.01)	0.972(0.01)
	dAiler	0.690(0.01)	0.721(0.02)	0.754(0.04)	0.723(0.02)	0.747(0.02)	0.742(0.04)	0.751(0.02)
	concreteStrength	0.775(0.03)	0.886(0.03)	0.871(0.03)	0.864(0.02)	0.870(0.03)	0.876(0.03)	0.872(0.02)
	acceleration	0.777(0.03)	0.869(0.02)	0.870(0.02)	0.856(0.02)	0.868(0.02)	0.873(0.03)	0.876(0.03)
	airfoild	0.458(0.04)	0.342(0.05)	0.346(0.03)	0.361(0.03)	0.351(0.03)	0.345(0.03)	0.353(0.04)

Table A.6: Evaluation results concerning the $NPval^{\phi}$ metric (average and standard deviation) for 20 data sets and all tested variants of 4 learners using unbiased pre-processing strategies.

Learner	Data sets	None	RU	RO	WERCS	SMOTER	GN	SMOGN
MARS	servo	0.968(0.04)	0.977(0.06)	0.970(0.05)	0.936(0.04)	0.967(0.03)	1.000(0.01)	1.000(0.01)
	a6	0.842(0.04)	0.766(0.13)	0.819(0.07)	0.799(0.06)	0.817(0.06)	0.801(0.06)	0.812(0.06)
	Abalone	0.896(0.01)	0.822(0.01)	0.823(0.01)	0.823(0.01)	0.801(0.01)	0.795(0.01)	0.794(0.01)
	machineCpu	0.977(0.02)	0.949(0.03)	0.958(0.02)	0.949(0.03)	0.961(0.02)	0.960(0.03)	0.955(0.02)
	a3	0.836(0.05)	0.743(0.09)	0.771(0.07)	0.784(0.07)	0.788(0.06)	0.787(0.06)	0.788(0.06)
	a4	0.869(0.05)	0.817(0.07)	0.812(0.06)	0.785(0.06)	0.820(0.06)	0.821(0.06)	0.825(0.06)
	a1	0.934(0.04)	0.863(0.06)	0.881(0.05)	0.863(0.05)	0.890(0.04)	0.880(0.04)	0.886(0.05)
	a7	0.873(0.05)	0.741(0.18)	0.799(0.08)	0.774(0.09)	0.819(0.10)	0.829(0.07)	0.818(0.08)
	boston	0.992(0.02)	0.957(0.02)	0.965(0.02)	0.959(0.02)	0.968(0.02)	0.961(0.02)	0.966(0.02)
	a2	0.914(0.04)	0.828(0.10)	0.838(0.05)	0.827(0.05)	0.839(0.06)	0.836(0.05)	0.846(0.05)
	a5	0.901(0.04)	0.774(0.10)	0.832(0.05)	0.814(0.06)	0.839(0.07)	0.840(0.06)	0.833(0.06)
	fuelCons	0.954(0.01)	0.934(0.01)	0.931(0.01)	0.941(0.01)	0.932(0.01)	0.922(0.01)	0.926(0.01)
	availPwr	0.988(0.00)	0.968(0.01)	0.973(0.01)	0.981(0.00)	0.971(0.01)	0.965(0.01)	0.967(0.01)
	cpuSm	0.854(0.01)	0.841(0.01)	0.840(0.01)	0.817(0.01)	0.846(0.01)	0.843(0.01)	0.845(0.01)
	maxTorque	0.994(0.00)	0.988(0.00)	0.974(0.01)	0.987(0.01)	0.976(0.01)	0.975(0.01)	0.978(0.01)
	bank8FM	0.995(0.00)	0.985(0.00)	0.984(0.00)	0.983(0.00)	0.986(0.00)	0.985(0.00)	0.985(0.00)
	dAiler	0.942(0.00)	0.859(0.01)	0.856(0.01)	0.887(0.01)	0.828(0.01)	0.824(0.01)	0.827(0.01)
	concreteStrength	0.998(0.01)	0.962(0.02)	0.969(0.01)	0.968(0.01)	0.980(0.01)	0.972(0.01)	0.977(0.01)
	acceleration	0.990(0.01)	0.967(0.01)	0.968(0.01)	0.962(0.01)	0.971(0.01)	0.966(0.01)	0.969(0.01)
	airfoild	0.835(0.02)	0.792(0.03)	0.801(0.02)	0.787(0.02)	0.816(0.02)	0.797(0.02)	0.804(0.02)
RF	servo	0.988(0.02)	1.000(0.02)	0.986(0.03)	0.979(0.02)	0.988(0.03)	1.000(0.01)	1.000(0.01)
	a6	0.835(0.04)	0.742(0.05)	0.811(0.04)	0.795(0.05)	0.805(0.06)	0.768(0.06)	0.802(0.06)
	Abalone	0.903(0.01)	0.842(0.01)	0.893(0.01)	0.867(0.01)	0.855(0.01)	0.850(0.01)	0.854(0.01)
	machineCpu	0.982(0.02)	0.950(0.05)	0.983(0.03)	0.972(0.02)	0.977(0.02)	0.978(0.02)	0.976(0.02)
	a3	0.838(0.05)	0.708(0.07)	0.823(0.04)	0.763(0.03)	0.781(0.06)	0.779(0.06)	0.796(0.07)
	a4	0.875(0.04)	0.744(0.06)	0.866(0.04)	0.818(0.06)	0.812(0.04)	0.812(0.06)	0.832(0.04)
	a1	0.929(0.03)	0.844(0.05)	0.928(0.04)	0.899(0.03)	0.899(0.04)	0.893(0.05)	0.905(0.04)

Continued on next page

Table A.6 – continued from previous page

Learner	Data sets	None	RU	RO	WERCS	SMOTER	GN	SMOGN
	a7	0.856(0.06)	0.666(0.09)	0.836(0.05)	0.764(0.05)	0.756(0.08)	0.753(0.07)	0.761(0.06)
	boston	0.992(0.01)	0.957(0.02)	0.989(0.01)	0.980(0.01)	0.984(0.02)	0.979(0.01)	0.985(0.02)
	a2	0.912(0.04)	0.757(0.06)	0.905(0.05)	0.852(0.05)	0.864(0.05)	0.850(0.05)	0.865(0.04)
	a5	0.900(0.04)	0.679(0.06)	0.886(0.04)	0.838(0.04)	0.847(0.05)	0.833(0.05)	0.840(0.04)
	fuelCons	0.979(0.00)	0.946(0.01)	0.979(0.01)	0.970(0.01)	0.963(0.01)	0.959(0.01)	0.961(0.01)
	availPwr	0.999(0.00)	0.981(0.01)	0.999(0.00)	0.998(0.00)	0.995(0.00)	0.996(0.00)	0.995(0.00)
RF	cpuSm	0.862(0.01)	0.855(0.01)	0.862(0.01)	0.849(0.01)	0.864(0.01)	0.863(0.01)	0.862(0.01)
	maxTorque	0.998(0.00)	0.973(0.01)	0.998(0.00)	0.998(0.00)	0.995(0.00)	0.993(0.00)	0.994(0.00)
	bank8FM	0.991(0.00)	0.979(0.00)	0.990(0.00)	0.987(0.00)	0.988(0.00)	0.987(0.00)	0.988(0.00)
	dAiler	0.946(0.00)	0.878(0.01)	0.942(0.00)	0.923(0.01)	0.906(0.01)	0.914(0.01)	0.909(0.01)
	concreteStrength	1.000(0.01)	0.966(0.02)	0.999(0.01)	0.991(0.01)	0.994(0.01)	0.995(0.01)	0.993(0.01)
	acceleration	0.996(0.01)	0.970(0.01)	0.995(0.00)	0.991(0.00)	0.989(0.01)	0.988(0.01)	0.988(0.01)
	airfoild	0.839(0.02)	0.811(0.02)	0.841(0.01)	0.835(0.01)	0.838(0.02)	0.840(0.02)	0.845(0.01)
	servo	0.909(0.05)	0.873(0.07)	0.884(0.07)	0.883(0.07)	0.878(0.06)	0.858(0.08)	0.879(0.07)
	a6	0.896(0.04)	0.819(0.06)	0.818(0.05)	0.803(0.05)	0.822(0.04)	0.821(0.05)	0.823(0.05)
	Abalone	0.913(0.01)	0.855(0.01)	0.858(0.01)	0.854(0.01)	0.838(0.01)	0.840(0.01)	0.836(0.01)
	machineCpu	0.974(0.01)	0.949(0.02)	0.957(0.02)	0.956(0.02)	0.959(0.02)	0.958(0.02)	0.959(0.02)
	a3	0.918(0.03)	0.787(0.07)	0.811(0.05)	0.787(0.08)	0.820(0.04)	0.790(0.06)	0.812(0.05)
	a4	0.895(0.05)	0.811(0.07)	0.833(0.05)	0.817(0.08)	0.831(0.06)	0.836(0.06)	0.841(0.05)
	a1	0.941(0.03)	0.830(0.07)	0.842(0.06)	0.842(0.06)	0.854(0.07)	0.849(0.06)	0.848(0.07)
	a7	0.915(0.03)	0.780(0.06)	0.798(0.06)	0.781(0.07)	0.816(0.07)	0.797(0.06)	0.789(0.06)
SVM	boston	0.985(0.01)	0.950(0.02)	0.962(0.01)	0.962(0.01)	0.959(0.01)	0.958(0.01)	0.959(0.01)
	a2	0.952(0.02)	0.801(0.07)	0.849(0.05)	0.813(0.06)	0.846(0.05)	0.840(0.05)	0.838(0.06)
	a5	0.947(0.03)	0.817(0.07)	0.838(0.05)	0.825(0.05)	0.844(0.05)	0.844(0.05)	0.848(0.06)
	fuelCons	0.970(0.01)	0.947(0.01)	0.965(0.01)	0.960(0.01)	0.954(0.01)	0.953(0.01)	0.953(0.01)
	availPwr	0.992(0.00)	0.979(0.01)	0.986(0.00)	0.988(0.00)	0.984(0.00)	0.980(0.00)	0.982(0.00)
	cpuSm	0.863(0.01)	0.800(0.01)	0.823(0.01)	0.823(0.01)	0.819(0.01)	0.817(0.01)	0.818(0.01)
	maxTorque	0.996(0.00)	0.984(0.00)	0.993(0.00)	0.993(0.00)	0.989(0.00)	0.988(0.00)	0.988(0.00)
	bank8FM	0.997(0.00)	0.990(0.00)	0.987(0.00)	0.984(0.00)	0.989(0.00)	0.987(0.00)	0.988(0.00)
	dAiler	0.956(0.00)	0.879(0.01)	0.878(0.01)	0.907(0.00)	0.851(0.01)	0.853(0.01)	0.852(0.01)

Continued on next page

Table A.6 – continued from previous page

Learner	Data sets	None	RU	RO	WERCS	SMOTER	GN	SMOGN
SVM	concreteStrength	0.992(0.01)	0.944(0.02)	0.955(0.01)	0.948(0.01)	0.955(0.01)	0.954(0.02)	0.954(0.01)
	acceleration	0.995(0.01)	0.957(0.01)	0.963(0.01)	0.968(0.01)	0.961(0.01)	0.962(0.01)	0.960(0.01)
	airfoild	0.853(0.02)	0.792(0.03)	0.821(0.02)	0.794(0.02)	0.823(0.02)	0.817(0.02)	0.819(0.02)
NNET	servo	0.934(0.03)	0.802(0.21)	0.942(0.12)	0.821(0.22)	0.848(0.25)	0.872(0.21)	0.799(0.22)
	a6	0.814(0.05)	0.488(0.26)	0.455(0.30)	0.434(0.27)	0.617(0.28)	0.514(0.30)	0.519(0.31)
	Abalone	0.910(0.01)	0.851(0.02)	0.857(0.02)	0.847(0.02)	0.847(0.02)	0.835(0.02)	0.842(0.02)
	machineCpu	0.844(0.04)	0.367(0.43)	0.231(0.28)	0.505(0.42)	0.484(0.42)	0.457(0.44)	0.438(0.40)
	a3	0.840(0.06)	0.630(0.19)	0.582(0.29)	0.530(0.24)	0.689(0.25)	0.633(0.30)	0.601(0.29)
	a4	0.869(0.06)	0.620(0.21)	0.585(0.25)	0.573(0.21)	0.706(0.20)	0.590(0.22)	0.647(0.22)
	a1	0.937(0.04)	0.792(0.11)	0.803(0.09)	0.704(0.09)	0.819(0.11)	0.816(0.11)	0.816(0.11)
	a7	0.843(0.10)	0.631(0.28)	0.625(0.26)	0.600(0.26)	0.683(0.28)	0.677(0.26)	0.702(0.27)
	boston	1.000(0.03)	0.777(0.10)	0.766(0.13)	0.741(0.15)	0.777(0.10)	0.783(0.12)	0.766(0.13)
	a2	0.913(0.04)	0.729(0.11)	0.738(0.12)	0.707(0.10)	0.785(0.13)	0.805(0.13)	0.767(0.13)
	a5	0.901(0.03)	0.713(0.21)	0.747(0.16)	0.730(0.13)	0.803(0.15)	0.827(0.15)	0.771(0.15)
	fuelCons	0.946(0.01)	0.850(0.05)	0.896(0.06)	0.906(0.04)	0.870(0.07)	0.872(0.07)	0.850(0.07)
	availPwr	1.000(0.02)	0.663(0.11)	0.604(0.21)	0.611(0.04)	0.606(0.07)	0.611(0.08)	0.603(0.06)
	cpuSm	0.823(0.06)	0.487(0.21)	0.474(0.17)	0.467(0.14)	0.485(0.23)	0.529(0.18)	0.554(0.22)
	maxTorque	1.000(0.01)	0.761(0.07)	0.713(0.09)	0.741(0.05)	0.722(0.07)	0.707(0.08)	0.728(0.08)
	bank8FM	0.991(0.00)	0.969(0.03)	0.979(0.02)	0.976(0.02)	0.985(0.01)	0.983(0.01)	0.983(0.02)
	dAiler	0.952(0.01)	0.884(0.03)	0.904(0.03)	0.915(0.01)	0.883(0.02)	0.885(0.02)	0.878(0.02)
	concreteStrength	1.000(0.01)	0.854(0.06)	0.806(0.07)	0.816(0.03)	0.808(0.05)	0.830(0.05)	0.817(0.05)
	acceleration	1.000(0.02)	0.760(0.06)	0.760(0.06)	0.801(0.04)	0.775(0.07)	0.784(0.07)	0.798(0.08)
	airfoild	0.894(0.02)	0.608(0.09)	0.579(0.08)	0.604(0.03)	0.581(0.07)	0.578(0.07)	0.603(0.09)

A.2 Evaluation Results of Biased Pre-processing Strategies

Table A.7: Evaluation results concerning the F_1^{ϕ} metric (average and standard deviation) for 20 data sets and all tested variants of 4 learners using biased pre-processing strategies.

Learner	Data sets	None	U._._	U.F._	U.S._	S._._	S.F.F	S.S.F	S.F.S	S.S.S
MARS	servo	0.645(0.11)	0.630(0.13)	0.628(0.13)	0.634(0.13)	0.657(0.12)	0.628(0.13)	0.645(0.12)	0.635(0.13)	0.646(0.11)
	a6	0.459(0.19)	0.517(0.10)	0.524(0.09)	0.509(0.11)	0.528(0.10)	0.539(0.09)	0.527(0.09)	0.528(0.11)	0.517(0.10)
	Abalone	0.708(0.02)	0.736(0.01)	0.736(0.02)	0.735(0.01)	0.733(0.01)	0.730(0.01)	0.731(0.01)	0.732(0.01)	0.734(0.01)
	machineCpu	0.797(0.09)	0.778(0.08)	0.757(0.11)	0.774(0.09)	0.780(0.09)	0.788(0.09)	0.783(0.08)	0.776(0.10)	0.782(0.08)
	a3	0.498(0.21)	0.533(0.09)	0.526(0.08)	0.516(0.09)	0.526(0.09)	0.520(0.08)	0.529(0.09)	0.532(0.10)	0.532(0.08)
	a4	0.481(0.20)	0.552(0.11)	0.525(0.12)	0.545(0.12)	0.552(0.10)	0.574(0.14)	0.586(0.10)	0.567(0.14)	0.584(0.10)
	a1	0.573(0.25)	0.720(0.10)	0.706(0.11)	0.722(0.10)	0.722(0.10)	0.724(0.09)	0.724(0.10)	0.722(0.09)	0.728(0.09)
	a7	0.294(0.17)	0.365(0.13)	0.355(0.15)	0.346(0.16)	0.361(0.14)	0.361(0.12)	0.340(0.14)	0.352(0.15)	0.360(0.13)
	boston	0.894(0.04)	0.873(0.05)	0.883(0.04)	0.878(0.04)	0.883(0.04)	0.880(0.04)	0.884(0.04)	0.880(0.04)	0.882(0.04)
	a2	0.260(0.38)	0.488(0.22)	0.527(0.20)	0.490(0.23)	0.522(0.20)	0.538(0.22)	0.537(0.21)	0.522(0.23)	0.524(0.22)
	a5	0.146(0.21)	0.495(0.22)	0.511(0.22)	0.526(0.20)	0.525(0.21)	0.527(0.20)	0.532(0.20)	0.511(0.20)	0.522(0.21)
	fuelCons	0.853(0.03)	0.865(0.03)	0.864(0.02)	0.867(0.03)	0.866(0.03)	0.862(0.02)	0.849(0.03)	0.875(0.02)	0.867(0.02)
	availPwr	0.902(0.02)	0.900(0.02)	0.903(0.02)	0.900(0.02)	0.898(0.02)	0.881(0.02)	0.878(0.02)	0.901(0.02)	0.901(0.02)
	cpuSm	0.142(0.03)	0.167(0.02)	0.170(0.03)	0.168(0.03)	0.170(0.02)	0.149(0.03)	0.149(0.03)	0.177(0.03)	0.177(0.03)
	maxTorque	0.954(0.01)	0.980(0.01)	0.979(0.01)	0.980(0.01)	0.959(0.02)	0.959(0.02)	0.958(0.02)	0.956(0.02)	0.956(0.02)
	bank8FM	0.943(0.01)	0.947(0.01)	0.947(0.01)	0.947(0.01)	0.947(0.01)	0.948(0.01)	0.947(0.01)	0.947(0.01)	0.947(0.01)
	dAiler	0.736(0.03)	0.754(0.02)	0.758(0.02)	0.753(0.02)	0.746(0.02)	0.745(0.02)	0.745(0.02)	0.750(0.02)	0.750(0.02)
	concreteStrength	0.886(0.03)	0.886(0.03)	0.882(0.04)	0.882(0.03)	0.898(0.03)	0.902(0.03)	0.899(0.03)	0.899(0.03)	0.896(0.03)
	acceleration	0.895(0.03)	0.886(0.04)	0.885(0.04)	0.890(0.03)	0.898(0.03)	0.898(0.02)	0.896(0.03)	0.894(0.03)	0.892(0.03)
	airfoild	0.116(0.08)	0.208(0.09)	0.202(0.10)	0.207(0.10)	0.214(0.11)	0.227(0.12)	0.207(0.11)	0.209(0.12)	0.204(0.13)
RF	servo	0.761(0.14)	0.730(0.12)	0.728(0.13)	0.723(0.13)	0.754(0.15)	0.732(0.15)	0.767(0.14)	0.740(0.14)	0.761(0.15)
	a6	0.527(0.14)	0.534(0.12)	0.545(0.12)	0.532(0.11)	0.524(0.14)	0.525(0.12)	0.525(0.12)	0.525(0.13)	0.526(0.11)
	Abalone	0.719(0.02)	0.726(0.02)	0.728(0.02)	0.725(0.02)	0.727(0.02)	0.727(0.01)	0.726(0.01)	0.726(0.02)	0.724(0.02)
	machineCpu	0.797(0.09)	0.788(0.09)	0.795(0.09)	0.782(0.07)	0.792(0.09)	0.788(0.08)	0.776(0.08)	0.785(0.09)	0.784(0.09)
	a3	0.453(0.20)	0.541(0.09)	0.541(0.07)	0.531(0.09)	0.566(0.09)	0.548(0.09)	0.555(0.09)	0.549(0.10)	0.558(0.10)

Continued on next page

Table A.7 – continued from previous page

Learner	Data sets	None	U._._	U.F._	U.S._	S._._	S.F.F	S.S.F	S.F.S	S.S.S
	a4	0.506(0.24)	0.585(0.12)	0.585(0.10)	0.587(0.10)	0.577(0.13)	0.587(0.10)	0.603(0.12)	0.577(0.14)	0.562(0.14)
	a1	0.621(0.29)	0.736(0.08)	0.746(0.10)	0.751(0.09)	0.727(0.09)	0.730(0.09)	0.731(0.08)	0.716(0.10)	0.731(0.09)
	a7	0.303(0.19)	0.382(0.15)	0.368(0.16)	0.371(0.16)	0.404(0.15)	0.370(0.15)	0.386(0.14)	0.364(0.16)	0.388(0.16)
	boston	0.902(0.04)	0.891(0.05)	0.897(0.05)	0.884(0.04)	0.896(0.04)	0.897(0.04)	0.901(0.05)	0.900(0.04)	0.900(0.05)
	a2	0.243(0.36)	0.576(0.22)	0.561(0.21)	0.573(0.22)	0.555(0.23)	0.588(0.23)	0.591(0.22)	0.541(0.25)	0.556(0.25)
	a5	0.209(0.30)	0.551(0.20)	0.553(0.21)	0.555(0.21)	0.548(0.22)	0.565(0.22)	0.541(0.21)	0.520(0.22)	0.530(0.23)
	fuelCons	0.918(0.01)	0.894(0.02)	0.898(0.02)	0.889(0.02)	0.920(0.02)	0.915(0.02)	0.910(0.01)	0.923(0.02)	0.920(0.01)
RF	availPwr	0.964(0.02)	0.951(0.02)	0.953(0.02)	0.948(0.02)	0.967(0.01)	0.966(0.02)	0.965(0.02)	0.967(0.01)	0.965(0.01)
	cpuSm	0.508(0.05)	0.484(0.06)	0.493(0.06)	0.480(0.06)	0.498(0.06)	0.487(0.05)	0.485(0.05)	0.508(0.05)	0.507(0.06)
	maxTorque	0.967(0.01)	0.949(0.02)	0.955(0.01)	0.945(0.02)	0.968(0.01)	0.970(0.01)	0.966(0.01)	0.971(0.01)	0.967(0.01)
	bank8FM	0.946(0.01)	0.941(0.01)	0.942(0.01)	0.941(0.01)	0.946(0.01)	0.946(0.01)	0.945(0.01)	0.949(0.01)	0.947(0.01)
	dAiler	0.735(0.03)	0.753(0.01)	0.757(0.02)	0.752(0.01)	0.746(0.02)	0.750(0.02)	0.745(0.02)	0.749(0.02)	0.748(0.02)
	concreteStrength	0.907(0.21)	0.907(0.03)	0.910(0.03)	0.902(0.04)	0.942(0.03)	0.928(0.09)	0.936(0.03)	0.911(0.12)	0.911(0.12)
	acceleration	0.934(0.02)	0.906(0.02)	0.914(0.03)	0.905(0.02)	0.941(0.02)	0.938(0.02)	0.935(0.02)	0.944(0.02)	0.940(0.02)
	airfoild	0.219(0.15)	0.214(0.09)	0.235(0.10)	0.228(0.12)	0.195(0.10)	0.219(0.09)	0.193(0.10)	0.191(0.11)	0.170(0.11)
	servo	0.366(0.12)	0.612(0.10)	0.635(0.10)	0.615(0.10)	0.648(0.10)	0.653(0.11)	0.648(0.10)	0.643(0.11)	0.634(0.10)
	a6	0.229(0.18)	0.526(0.11)	0.530(0.11)	0.534(0.09)	0.522(0.10)	0.535(0.10)	0.545(0.10)	0.505(0.09)	0.522(0.10)
	Abalone	0.712(0.03)	0.735(0.02)	0.734(0.02)	0.733(0.02)	0.734(0.01)	0.733(0.01)	0.733(0.01)	0.733(0.02)	0.734(0.01)
	machineCpu	0.780(0.10)	0.785(0.07)	0.779(0.08)	0.782(0.08)	0.782(0.08)	0.791(0.08)	0.789(0.07)	0.787(0.08)	0.778(0.09)
	a3	0.181(0.10)	0.520(0.10)	0.525(0.10)	0.520(0.09)	0.532(0.10)	0.543(0.09)	0.539(0.09)	0.534(0.10)	0.521(0.10)
	a4	0.252(0.24)	0.568(0.10)	0.557(0.10)	0.547(0.11)	0.578(0.12)	0.574(0.15)	0.574(0.11)	0.570(0.15)	0.579(0.11)
	a1	0.113(0.18)	0.690(0.08)	0.677(0.08)	0.698(0.07)	0.715(0.09)	0.715(0.09)	0.726(0.08)	0.704(0.10)	0.714(0.09)
SVM	a7	0.107(0.10)	0.325(0.16)	0.342(0.14)	0.316(0.16)	0.343(0.15)	0.353(0.14)	0.349(0.15)	0.331(0.15)	0.339(0.16)
	boston	0.883(0.04)	0.878(0.04)	0.878(0.04)	0.878(0.04)	0.896(0.04)	0.899(0.04)	0.898(0.04)	0.895(0.04)	0.893(0.04)
	a2	0.232(0.34)	0.494(0.22)	0.478(0.22)	0.498(0.24)	0.504(0.24)	0.515(0.24)	0.506(0.25)	0.458(0.27)	0.456(0.26)
	a5	0.139(0.17)	0.534(0.21)	0.530(0.21)	0.530(0.22)	0.547(0.22)	0.558(0.22)	0.548(0.21)	0.529(0.23)	0.498(0.24)
	fuelCons	0.908(0.02)	0.880(0.03)	0.884(0.03)	0.876(0.03)	0.892(0.02)	0.889(0.02)	0.888(0.02)	0.899(0.02)	0.896(0.02)
	availPwr	0.935(0.01)	0.924(0.02)	0.928(0.02)	0.924(0.02)	0.938(0.01)	0.922(0.02)	0.922(0.02)	0.938(0.01)	0.938(0.01)
	cpuSm	0.161(0.03)	0.166(0.03)	0.168(0.03)	0.168(0.03)	0.179(0.03)	0.176(0.03)	0.172(0.03)	0.185(0.03)	0.183(0.03)
	maxTorque	0.973(0.01)	0.958(0.02)	0.965(0.01)	0.960(0.02)	0.972(0.01)	0.970(0.01)	0.969(0.01)	0.971(0.01)	0.970(0.01)

Continued on next page

Table A.7 – continued from previous page

Learner	Data sets	None	U._._	U.F._	U.S._	S._._	S.F.F	S.S.F	S.F.S	S.S.S
SVM	bank8FM	0.947(0.01)	0.949(0.01)	0.949(0.01)	0.948(0.01)	0.950(0.01)	0.949(0.01)	0.949(0.01)	0.950(0.01)	0.950(0.01)
	dAiler	0.728(0.03)	0.761(0.02)	0.762(0.02)	0.759(0.02)	0.751(0.01)	0.747(0.01)	0.748(0.01)	0.758(0.02)	0.757(0.01)
	concreteStrength	0.840(0.14)	0.875(0.04)	0.888(0.04)	0.879(0.04)	0.909(0.03)	0.909(0.03)	0.909(0.03)	0.911(0.03)	0.911(0.03)
	acceleration	0.872(0.07)	0.898(0.03)	0.900(0.04)	0.895(0.04)	0.899(0.02)	0.895(0.02)	0.896(0.02)	0.906(0.02)	0.905(0.02)
	airfoild	0.158(0.09)	0.216(0.12)	0.210(0.11)	0.221(0.11)	0.240(0.11)	0.218(0.09)	0.226(0.10)	0.245(0.12)	0.249(0.12)
NNET	servo	0.654(0.14)	0.619(0.18)	0.667(0.18)	0.641(0.18)	0.666(0.18)	0.664(0.19)	0.696(0.20)	0.664(0.18)	0.685(0.17)
	a6	0.176(0.19)	0.515(0.11)	0.518(0.11)	0.508(0.12)	0.518(0.11)	0.509(0.10)	0.509(0.11)	0.501(0.11)	0.507(0.11)
	Abalone	0.693(0.06)	0.704(0.08)	0.708(0.07)	0.706(0.07)	0.708(0.08)	0.716(0.07)	0.717(0.07)	0.711(0.08)	0.711(0.08)
	machineCpu	0.062(0.08)	0.531(0.10)	0.537(0.08)	0.540(0.09)	0.555(0.09)	0.396(0.27)	0.308(0.28)	0.552(0.09)	0.542(0.08)
	a3	0.197(0.21)	0.507(0.08)	0.508(0.08)	0.497(0.08)	0.513(0.08)	0.510(0.07)	0.502(0.08)	0.512(0.08)	0.502(0.10)
	a4	0.422(0.25)	0.504(0.15)	0.487(0.17)	0.499(0.13)	0.548(0.14)	0.548(0.13)	0.554(0.13)	0.564(0.13)	0.567(0.14)
	a1	0.243(0.26)	0.433(0.24)	0.484(0.19)	0.462(0.22)	0.416(0.26)	0.345(0.25)	0.368(0.24)	0.394(0.28)	0.399(0.27)
	a7	0.263(0.23)	0.347(0.14)	0.338(0.14)	0.362(0.15)	0.359(0.14)	0.345(0.15)	0.360(0.15)	0.359(0.15)	0.358(0.15)
	boston	0.311(0.29)	0.355(0.21)	0.376(0.22)	0.369(0.22)	0.298(0.24)	0.351(0.23)	0.345(0.26)	0.332(0.26)	0.346(0.24)
	a2	0.044(0.13)	0.334(0.22)	0.356(0.25)	0.376(0.22)	0.374(0.26)	0.364(0.27)	0.361(0.28)	0.378(0.26)	0.354(0.27)
	a5	0.011(0.04)	0.399(0.23)	0.383(0.23)	0.388(0.23)	0.378(0.24)	0.385(0.25)	0.377(0.25)	0.415(0.25)	0.405(0.25)
	fuelCons	0.056(0.13)	0.375(0.20)	0.354(0.23)	0.341(0.23)	0.302(0.24)	0.260(0.23)	0.277(0.25)	0.349(0.19)	0.347(0.22)
	availPwr	0.096(0.12)	0.180(0.21)	0.152(0.17)	0.157(0.17)	0.060(0.14)	0.052(0.14)	0.045(0.13)	0.041(0.12)	0.038(0.11)
	cpuSm	0.084(0.07)	0.246(0.17)	0.218(0.16)	0.241(0.16)	0.230(0.17)	0.194(0.16)	0.201(0.16)	0.195(0.18)	0.220(0.17)
	maxTorque	0.088(0.17)	0.443(0.22)	0.412(0.26)	0.437(0.23)	0.270(0.18)	0.232(0.26)	0.171(0.20)	0.302(0.27)	0.311(0.25)
	bank8FM	0.953(0.01)	0.898(0.10)	0.898(0.10)	0.898(0.10)	0.946(0.03)	0.920(0.06)	0.927(0.05)	0.924(0.10)	0.935(0.07)
	dAiler	0.329(0.08)	0.332(0.03)	0.334(0.03)	0.340(0.06)	0.374(0.06)	0.389(0.10)	0.390(0.10)	0.398(0.11)	0.385(0.10)
	concreteStrength	0.000(0.00)	0.293(0.23)	0.245(0.21)	0.296(0.19)	0.127(0.19)	0.097(0.19)	0.132(0.20)	0.214(0.25)	0.212(0.26)
	acceleration	0.099(0.24)	0.310(0.22)	0.300(0.22)	0.373(0.23)	0.259(0.25)	0.271(0.28)	0.273(0.28)	0.386(0.25)	0.382(0.26)
	airfoild	0.020(0.03)	0.070(0.06)	0.069(0.06)	0.070(0.06)	0.029(0.04)	0.032(0.05)	0.028(0.04)	0.044(0.06)	0.038(0.05)

Table A.8: Evaluation results concerning the $G-Mean^{\phi}$ metric (average and standard deviation) for 20 data sets and all tested variants of 4 learners using biased pre-processing strategies.

Learner	Data sets	None	U._._	U.F._	U.S._	S._._	S.F.F	S.S.F	S.F.S	S.S.S
MARS	servo	0.734(0.12)	0.738(0.12)	0.735(0.12)	0.741(0.13)	0.759(0.12)	0.737(0.12)	0.749(0.12)	0.739(0.12)	0.745(0.11)
	a6	0.596(0.11)	0.629(0.14)	0.644(0.13)	0.629(0.14)	0.642(0.13)	0.653(0.11)	0.642(0.12)	0.637(0.13)	0.631(0.12)
	Abalone	0.734(0.02)	0.803(0.02)	0.796(0.02)	0.805(0.02)	0.813(0.02)	0.815(0.02)	0.814(0.02)	0.808(0.02)	0.807(0.02)
	machineCpu	0.845(0.07)	0.851(0.07)	0.828(0.09)	0.850(0.08)	0.847(0.08)	0.857(0.08)	0.860(0.07)	0.849(0.06)	0.848(0.06)
	a3	0.601(0.12)	0.679(0.11)	0.665(0.11)	0.671(0.13)	0.663(0.10)	0.671(0.11)	0.669(0.11)	0.665(0.11)	0.667(0.10)
	a4	0.638(0.11)	0.710(0.14)	0.672(0.15)	0.696(0.13)	0.689(0.12)	0.696(0.15)	0.714(0.13)	0.693(0.15)	0.717(0.11)
	a1	0.757(0.10)	0.825(0.09)	0.820(0.08)	0.823(0.09)	0.815(0.09)	0.822(0.10)	0.828(0.10)	0.813(0.09)	0.823(0.09)
	a7	0.440(0.18)	0.497(0.20)	0.473(0.21)	0.462(0.22)	0.484(0.21)	0.476(0.18)	0.451(0.20)	0.469(0.23)	0.479(0.19)
	boston	0.914(0.05)	0.923(0.04)	0.923(0.04)	0.924(0.04)	0.924(0.04)	0.924(0.04)	0.926(0.04)	0.917(0.05)	0.920(0.04)
	a2	0.623(0.25)	0.679(0.25)	0.675(0.25)	0.664(0.27)	0.660(0.25)	0.677(0.27)	0.676(0.27)	0.663(0.26)	0.668(0.26)
	a5	0.587(0.23)	0.667(0.26)	0.681(0.26)	0.676(0.26)	0.663(0.26)	0.651(0.26)	0.670(0.26)	0.637(0.26)	0.657(0.26)
	fuelCons	0.878(0.02)	0.910(0.02)	0.911(0.02)	0.914(0.02)	0.913(0.02)	0.915(0.02)	0.907(0.02)	0.914(0.01)	0.911(0.02)
	availPwr	0.925(0.02)	0.934(0.01)	0.935(0.01)	0.935(0.01)	0.933(0.01)	0.923(0.02)	0.921(0.01)	0.934(0.01)	0.933(0.01)
	cpuSm	0.220(0.03)	0.247(0.03)	0.251(0.03)	0.248(0.03)	0.250(0.03)	0.228(0.04)	0.228(0.04)	0.259(0.03)	0.259(0.03)
	maxTorque	0.964(0.01)	0.988(0.00)	0.986(0.01)	0.988(0.00)	0.973(0.01)	0.971(0.01)	0.972(0.01)	0.972(0.01)	0.971(0.01)
	bank8FM	0.950(0.01)	0.957(0.01)	0.956(0.01)	0.957(0.01)	0.957(0.01)	0.958(0.01)	0.957(0.01)	0.955(0.01)	0.955(0.01)
	dAiler	0.748(0.02)	0.849(0.02)	0.844(0.02)	0.850(0.02)	0.859(0.02)	0.862(0.02)	0.861(0.02)	0.848(0.02)	0.847(0.02)
	concreteStrength	0.904(0.03)	0.942(0.03)	0.944(0.03)	0.947(0.02)	0.944(0.03)	0.945(0.02)	0.946(0.02)	0.942(0.02)	0.941(0.03)
	acceleration	0.919(0.02)	0.948(0.02)	0.946(0.02)	0.947(0.02)	0.946(0.02)	0.946(0.01)	0.945(0.02)	0.942(0.02)	0.941(0.02)
	airfoild	0.229(0.07)	0.283(0.10)	0.272(0.12)	0.283(0.12)	0.294(0.12)	0.308(0.13)	0.288(0.11)	0.286(0.12)	0.277(0.12)
RF	servo	0.816(0.12)	0.813(0.11)	0.807(0.12)	0.808(0.12)	0.822(0.14)	0.810(0.13)	0.827(0.12)	0.818(0.13)	0.824(0.13)
	a6	0.600(0.15)	0.664(0.17)	0.667(0.16)	0.660(0.16)	0.645(0.17)	0.632(0.16)	0.645(0.14)	0.635(0.17)	0.648(0.15)
	Abalone	0.748(0.02)	0.808(0.02)	0.801(0.02)	0.809(0.02)	0.802(0.02)	0.806(0.02)	0.805(0.02)	0.796(0.02)	0.795(0.02)
	machineCpu	0.853(0.07)	0.856(0.07)	0.853(0.07)	0.856(0.06)	0.855(0.07)	0.844(0.06)	0.844(0.06)	0.847(0.07)	0.852(0.07)
	a3	0.606(0.11)	0.693(0.12)	0.679(0.10)	0.682(0.12)	0.691(0.11)	0.676(0.11)	0.690(0.11)	0.669(0.11)	0.681(0.11)
	a4	0.662(0.11)	0.729(0.14)	0.732(0.11)	0.739(0.12)	0.703(0.11)	0.714(0.11)	0.717(0.11)	0.709(0.11)	0.714(0.10)
	a1	0.772(0.10)	0.834(0.09)	0.827(0.10)	0.845(0.09)	0.809(0.10)	0.807(0.10)	0.822(0.10)	0.800(0.10)	0.814(0.10)

Continued on next page

Table A.8 – continued from previous page

Learner	Data sets	None	U._._	U.F._	U.S._	S._._	S.F.F	S.S.F	S.F.S	S.S.S
	a7	0.450(0.18)	0.504(0.23)	0.482(0.23)	0.490(0.23)	0.535(0.21)	0.481(0.21)	0.497(0.20)	0.481(0.22)	0.501(0.21)
	boston	0.920(0.05)	0.930(0.04)	0.928(0.04)	0.930(0.04)	0.928(0.04)	0.930(0.05)	0.931(0.04)	0.928(0.05)	0.927(0.04)
	a2	0.615(0.24)	0.711(0.27)	0.690(0.26)	0.705(0.27)	0.675(0.25)	0.691(0.27)	0.692(0.26)	0.674(0.26)	0.674(0.25)
	a5	0.604(0.24)	0.692(0.26)	0.700(0.26)	0.699(0.27)	0.660(0.26)	0.662(0.26)	0.667(0.26)	0.646(0.26)	0.651(0.26)
	fuelCons	0.922(0.01)	0.931(0.02)	0.931(0.02)	0.931(0.02)	0.942(0.01)	0.938(0.01)	0.940(0.02)	0.938(0.01)	0.940(0.01)
	availPwr	0.973(0.01)	0.972(0.01)	0.972(0.01)	0.972(0.01)	0.977(0.01)	0.976(0.01)	0.976(0.01)	0.976(0.01)	0.976(0.01)
RF	cpuSm	0.566(0.05)	0.584(0.06)	0.580(0.05)	0.580(0.05)	0.581(0.05)	0.576(0.05)	0.576(0.05)	0.582(0.05)	0.584(0.06)
	maxTorque	0.974(0.01)	0.975(0.01)	0.975(0.01)	0.975(0.01)	0.980(0.01)	0.980(0.01)	0.980(0.01)	0.979(0.01)	0.979(0.01)
	bank8FM	0.955(0.01)	0.964(0.01)	0.964(0.01)	0.965(0.00)	0.965(0.00)	0.964(0.01)	0.964(0.01)	0.963(0.00)	0.964(0.00)
	dAiler	0.770(0.02)	0.861(0.02)	0.854(0.02)	0.865(0.02)	0.843(0.02)	0.846(0.02)	0.844(0.02)	0.832(0.02)	0.830(0.02)
	concreteStrength	0.942(0.02)	0.951(0.02)	0.949(0.02)	0.950(0.02)	0.952(0.02)	0.954(0.02)	0.957(0.02)	0.949(0.02)	0.951(0.02)
	acceleration	0.948(0.02)	0.959(0.01)	0.957(0.01)	0.958(0.01)	0.965(0.01)	0.960(0.01)	0.960(0.01)	0.962(0.01)	0.962(0.01)
	airfoild	0.287(0.14)	0.298(0.10)	0.332(0.11)	0.304(0.12)	0.278(0.11)	0.310(0.10)	0.277(0.12)	0.267(0.11)	0.245(0.10)
	servo	0.497(0.14)	0.744(0.12)	0.756(0.11)	0.755(0.12)	0.778(0.11)	0.775(0.12)	0.777(0.12)	0.763(0.12)	0.762(0.12)
	a6	0.550(0.10)	0.640(0.14)	0.640(0.12)	0.647(0.11)	0.634(0.12)	0.635(0.12)	0.647(0.12)	0.613(0.11)	0.635(0.12)
	Abalone	0.717(0.02)	0.794(0.02)	0.782(0.02)	0.795(0.02)	0.809(0.02)	0.812(0.02)	0.811(0.02)	0.800(0.02)	0.800(0.02)
	machineCpu	0.839(0.08)	0.851(0.07)	0.843(0.07)	0.850(0.07)	0.849(0.07)	0.856(0.07)	0.858(0.07)	0.850(0.07)	0.845(0.07)
	a3	0.536(0.07)	0.671(0.12)	0.661(0.11)	0.660(0.12)	0.662(0.12)	0.680(0.12)	0.671(0.11)	0.658(0.12)	0.646(0.12)
	a4	0.597(0.10)	0.696(0.13)	0.678(0.13)	0.692(0.11)	0.682(0.13)	0.716(0.12)	0.695(0.12)	0.695(0.12)	0.682(0.13)
	a1	0.679(0.06)	0.815(0.09)	0.806(0.08)	0.827(0.08)	0.822(0.09)	0.830(0.09)	0.828(0.09)	0.813(0.09)	0.815(0.09)
	a7	0.434(0.14)	0.423(0.22)	0.440(0.20)	0.414(0.22)	0.452(0.21)	0.466(0.19)	0.462(0.21)	0.435(0.20)	0.437(0.22)
SVM	boston	0.900(0.04)	0.923(0.04)	0.922(0.04)	0.924(0.03)	0.930(0.04)	0.935(0.03)	0.935(0.03)	0.925(0.04)	0.925(0.04)
	a2	0.591(0.24)	0.655(0.26)	0.634(0.25)	0.644(0.26)	0.665(0.26)	0.652(0.26)	0.656(0.26)	0.629(0.25)	0.626(0.24)
	a5	0.558(0.21)	0.674(0.27)	0.671(0.26)	0.678(0.27)	0.665(0.26)	0.664(0.26)	0.665(0.26)	0.657(0.26)	0.654(0.26)
	fuelCons	0.921(0.02)	0.927(0.02)	0.926(0.02)	0.925(0.02)	0.927(0.02)	0.925(0.02)	0.926(0.02)	0.923(0.02)	0.925(0.02)
	availPwr	0.950(0.01)	0.955(0.01)	0.955(0.01)	0.955(0.01)	0.957(0.01)	0.949(0.01)	0.949(0.01)	0.957(0.01)	0.957(0.01)
	cpuSm	0.241(0.03)	0.238(0.04)	0.241(0.04)	0.241(0.04)	0.262(0.04)	0.260(0.04)	0.256(0.04)	0.264(0.04)	0.262(0.04)
	maxTorque	0.979(0.01)	0.980(0.00)	0.980(0.00)	0.979(0.00)	0.980(0.01)	0.978(0.01)	0.978(0.01)	0.979(0.01)	0.979(0.01)
	bank8FM	0.955(0.01)	0.955(0.01)	0.955(0.01)	0.955(0.01)	0.958(0.01)	0.959(0.01)	0.958(0.01)	0.957(0.01)	0.956(0.01)
	dAiler	0.740(0.02)	0.848(0.02)	0.843(0.01)	0.849(0.01)	0.864(0.01)	0.867(0.01)	0.867(0.01)	0.851(0.02)	0.850(0.02)

Continued on next page

Table A.8 – continued from previous page

Learner	Data sets	None	U._._	U.F._	U.S._	S._._	S.F.F	S.S.F	S.F.S	S.S.S
SVM	concreteStrength	0.899(0.03)	0.943(0.02)	0.944(0.02)	0.946(0.02)	0.953(0.02)	0.953(0.02)	0.953(0.02)	0.950(0.02)	0.949(0.02)
	acceleration	0.907(0.02)	0.949(0.02)	0.946(0.02)	0.948(0.02)	0.946(0.02)	0.947(0.01)	0.948(0.01)	0.941(0.02)	0.941(0.02)
	airfoild	0.237(0.09)	0.294(0.12)	0.287(0.12)	0.296(0.11)	0.318(0.11)	0.296(0.10)	0.302(0.10)	0.325(0.12)	0.327(0.12)
NNET	servo	0.746(0.13)	0.708(0.17)	0.748(0.16)	0.728(0.17)	0.738(0.17)	0.737(0.18)	0.762(0.18)	0.739(0.16)	0.754(0.16)
	a6	0.562(0.12)	0.637(0.16)	0.649(0.15)	0.647(0.15)	0.641(0.16)	0.629(0.15)	0.632(0.15)	0.626(0.15)	0.635(0.15)
	Abalone	0.736(0.03)	0.798(0.04)	0.790(0.04)	0.800(0.04)	0.808(0.05)	0.818(0.04)	0.818(0.04)	0.802(0.04)	0.801(0.04)
	machineCpu	0.582(0.11)	0.689(0.14)	0.697(0.14)	0.693(0.14)	0.700(0.14)	0.686(0.14)	0.674(0.14)	0.705(0.13)	0.690(0.13)
	a3	0.559(0.09)	0.656(0.13)	0.650(0.12)	0.644(0.11)	0.648(0.11)	0.652(0.11)	0.644(0.11)	0.646(0.11)	0.642(0.11)
	a4	0.635(0.12)	0.702(0.13)	0.702(0.13)	0.689(0.13)	0.715(0.12)	0.709(0.12)	0.717(0.12)	0.719(0.12)	0.719(0.13)
	a1	0.694(0.10)	0.779(0.09)	0.775(0.09)	0.780(0.09)	0.787(0.08)	0.776(0.09)	0.780(0.08)	0.781(0.10)	0.777(0.09)
	a7	0.468(0.21)	0.472(0.20)	0.459(0.20)	0.487(0.21)	0.487(0.20)	0.464(0.21)	0.490(0.22)	0.488(0.21)	0.484(0.22)
	boston	0.748(0.07)	0.821(0.05)	0.828(0.05)	0.819(0.05)	0.810(0.05)	0.818(0.05)	0.814(0.05)	0.815(0.06)	0.816(0.06)
	a2	0.574(0.22)	0.675(0.26)	0.664(0.25)	0.684(0.26)	0.671(0.26)	0.679(0.26)	0.682(0.26)	0.669(0.25)	0.669(0.25)
	a5	0.540(0.21)	0.664(0.26)	0.660(0.26)	0.666(0.26)	0.659(0.26)	0.656(0.26)	0.660(0.26)	0.662(0.26)	0.661(0.26)
	fuelCons	0.518(0.04)	0.626(0.05)	0.639(0.05)	0.621(0.06)	0.609(0.08)	0.607(0.07)	0.607(0.08)	0.618(0.07)	0.625(0.07)
	availPwr	0.637(0.05)	0.776(0.05)	0.772(0.05)	0.769(0.05)	0.755(0.05)	0.743(0.04)	0.742(0.04)	0.756(0.05)	0.753(0.04)
	cpuSm	0.241(0.06)	0.279(0.13)	0.253(0.13)	0.277(0.13)	0.257(0.13)	0.262(0.11)	0.274(0.12)	0.240(0.14)	0.263(0.14)
	maxTorque	0.667(0.04)	0.809(0.04)	0.813(0.04)	0.803(0.04)	0.797(0.04)	0.798(0.04)	0.788(0.04)	0.802(0.04)	0.796(0.04)
	bank8FM	0.962(0.01)	0.956(0.02)	0.956(0.02)	0.956(0.02)	0.965(0.01)	0.960(0.01)	0.961(0.01)	0.961(0.02)	0.963(0.02)
	dAiler	0.607(0.02)	0.653(0.03)	0.655(0.03)	0.654(0.03)	0.688(0.03)	0.687(0.04)	0.685(0.04)	0.690(0.04)	0.686(0.04)
	concreteStrength	0.700(0.03)	0.856(0.04)	0.861(0.03)	0.856(0.04)	0.832(0.04)	0.830(0.04)	0.831(0.04)	0.848(0.04)	0.843(0.04)
	acceleration	0.699(0.04)	0.834(0.03)	0.842(0.03)	0.840(0.03)	0.831(0.03)	0.822(0.04)	0.824(0.04)	0.838(0.04)	0.836(0.03)
	airfoild	0.341(0.04)	0.241(0.05)	0.229(0.05)	0.244(0.05)	0.249(0.03)	0.246(0.04)	0.252(0.03)	0.238(0.05)	0.243(0.04)

Table A.9: Evaluation results concerning the $prec^{\phi}$ metric (average and standard deviation) for 20 data sets and all tested variants of 4 learners using biased pre-processing strategies.

Learner	Data sets	None	U._._	U.F._	U.S._	S._._	S.F.F	S.S.F	S.F.S	S.S.S
MARS	servo	0.614(0.10)	0.585(0.13)	0.583(0.13)	0.588(0.13)	0.611(0.12)	0.581(0.13)	0.599(0.12)	0.591(0.13)	0.607(0.11)
	a6	0.475(0.20)	0.485(0.06)	0.480(0.06)	0.467(0.07)	0.491(0.07)	0.500(0.06)	0.489(0.06)	0.497(0.08)	0.484(0.07)
	Abalone	0.731(0.03)	0.698(0.02)	0.707(0.02)	0.693(0.01)	0.679(0.01)	0.673(0.01)	0.675(0.01)	0.684(0.01)	0.688(0.01)
	machineCpu	0.794(0.11)	0.748(0.09)	0.737(0.12)	0.742(0.10)	0.759(0.10)	0.764(0.10)	0.751(0.10)	0.754(0.12)	0.762(0.10)
	a3	0.531(0.25)	0.463(0.10)	0.461(0.07)	0.438(0.05)	0.467(0.08)	0.449(0.07)	0.465(0.08)	0.473(0.09)	0.472(0.08)
	a4	0.482(0.25)	0.478(0.09)	0.467(0.10)	0.484(0.11)	0.498(0.09)	0.528(0.13)	0.528(0.09)	0.518(0.13)	0.524(0.09)
	a1	0.573(0.28)	0.668(0.11)	0.651(0.12)	0.668(0.11)	0.680(0.11)	0.677(0.10)	0.669(0.11)	0.683(0.11)	0.683(0.10)
	a7	0.307(0.15)	0.338(0.06)	0.346(0.06)	0.344(0.06)	0.346(0.05)	0.343(0.05)	0.335(0.05)	0.347(0.05)	0.342(0.05)
	boston	0.898(0.04)	0.847(0.06)	0.866(0.05)	0.852(0.06)	0.862(0.05)	0.859(0.05)	0.863(0.05)	0.867(0.05)	0.866(0.05)
	a2	0.308(0.41)	0.462(0.13)	0.486(0.10)	0.463(0.12)	0.505(0.08)	0.518(0.10)	0.521(0.09)	0.514(0.13)	0.512(0.11)
	a5	0.156(0.22)	0.466(0.12)	0.486(0.11)	0.495(0.07)	0.520(0.10)	0.528(0.06)	0.519(0.08)	0.513(0.09)	0.514(0.10)
	fuelCons	0.851(0.05)	0.835(0.05)	0.833(0.04)	0.837(0.05)	0.833(0.04)	0.823(0.03)	0.807(0.04)	0.850(0.04)	0.837(0.04)
	availPwr	0.900(0.02)	0.884(0.03)	0.888(0.02)	0.884(0.03)	0.882(0.03)	0.860(0.02)	0.858(0.02)	0.887(0.03)	0.887(0.02)
	cpuSm	0.144(0.03)	0.168(0.02)	0.171(0.02)	0.169(0.02)	0.170(0.02)	0.149(0.02)	0.149(0.02)	0.178(0.03)	0.177(0.03)
	maxTorque	0.954(0.02)	0.974(0.02)	0.975(0.02)	0.975(0.02)	0.953(0.02)	0.954(0.02)	0.953(0.02)	0.948(0.03)	0.949(0.03)
	bank8FM	0.948(0.01)	0.948(0.01)	0.947(0.01)	0.948(0.01)	0.948(0.01)	0.947(0.01)	0.947(0.01)	0.950(0.01)	0.950(0.01)
	dAiler	0.786(0.04)	0.697(0.02)	0.709(0.02)	0.694(0.02)	0.673(0.02)	0.669(0.02)	0.669(0.02)	0.691(0.02)	0.691(0.02)
	concreteStrength	0.895(0.06)	0.847(0.05)	0.838(0.06)	0.834(0.05)	0.868(0.05)	0.873(0.05)	0.869(0.05)	0.872(0.05)	0.868(0.05)
	acceleration	0.892(0.05)	0.843(0.06)	0.843(0.06)	0.849(0.05)	0.865(0.04)	0.865(0.04)	0.863(0.04)	0.864(0.04)	0.861(0.05)
	airfoild	0.118(0.09)	0.215(0.09)	0.215(0.10)	0.217(0.10)	0.218(0.12)	0.229(0.12)	0.211(0.11)	0.218(0.14)	0.220(0.17)
RF	servo	0.750(0.14)	0.691(0.12)	0.695(0.13)	0.683(0.13)	0.730(0.15)	0.701(0.14)	0.749(0.14)	0.707(0.14)	0.741(0.16)
	a6	0.549(0.14)	0.482(0.07)	0.499(0.06)	0.482(0.07)	0.490(0.10)	0.502(0.08)	0.488(0.09)	0.494(0.09)	0.484(0.08)
	Abalone	0.736(0.03)	0.675(0.02)	0.686(0.02)	0.672(0.02)	0.682(0.02)	0.678(0.02)	0.677(0.01)	0.687(0.02)	0.686(0.02)
	machineCpu	0.783(0.09)	0.761(0.10)	0.775(0.09)	0.750(0.08)	0.770(0.09)	0.774(0.08)	0.752(0.09)	0.766(0.10)	0.759(0.10)
	a3	0.459(0.22)	0.460(0.06)	0.472(0.06)	0.452(0.06)	0.511(0.08)	0.495(0.10)	0.493(0.09)	0.507(0.10)	0.508(0.09)
	a4	0.496(0.27)	0.516(0.10)	0.513(0.09)	0.512(0.08)	0.530(0.14)	0.534(0.10)	0.562(0.13)	0.534(0.16)	0.505(0.15)
	a1	0.644(0.32)	0.683(0.07)	0.709(0.10)	0.698(0.09)	0.696(0.09)	0.707(0.11)	0.693(0.09)	0.688(0.11)	0.699(0.10)

Continued on next page

Table A.9 – continued from previous page

Learner	Data sets	None	U._._	U.F._	U.S._	S._._	S.F.F	S.S.F	S.F.S	S.S.S
	a7	0.314(0.18)	0.365(0.04)	0.360(0.05)	0.364(0.05)	0.379(0.06)	0.365(0.06)	0.377(0.05)	0.365(0.06)	0.380(0.06)
	boston	0.909(0.04)	0.873(0.05)	0.885(0.05)	0.858(0.05)	0.884(0.04)	0.884(0.04)	0.888(0.04)	0.892(0.04)	0.892(0.05)
	a2	0.291(0.38)	0.551(0.09)	0.558(0.09)	0.555(0.07)	0.561(0.14)	0.602(0.12)	0.614(0.11)	0.554(0.17)	0.582(0.16)
	a5	0.244(0.32)	0.518(0.05)	0.524(0.07)	0.528(0.06)	0.554(0.11)	0.589(0.09)	0.538(0.10)	0.528(0.14)	0.541(0.15)
	fuelCons	0.928(0.02)	0.864(0.03)	0.871(0.04)	0.854(0.03)	0.904(0.03)	0.899(0.03)	0.887(0.03)	0.915(0.03)	0.905(0.03)
	availPwr	0.962(0.02)	0.936(0.03)	0.941(0.02)	0.931(0.03)	0.962(0.02)	0.963(0.02)	0.962(0.02)	0.965(0.01)	0.961(0.02)
RF	cpuSm	0.525(0.05)	0.455(0.06)	0.476(0.05)	0.451(0.05)	0.486(0.06)	0.470(0.05)	0.466(0.05)	0.505(0.05)	0.502(0.06)
	maxTorque	0.969(0.01)	0.928(0.03)	0.940(0.02)	0.921(0.03)	0.961(0.02)	0.966(0.01)	0.957(0.02)	0.968(0.01)	0.960(0.02)
	bank8FM	0.947(0.01)	0.926(0.01)	0.928(0.01)	0.925(0.01)	0.936(0.01)	0.937(0.01)	0.933(0.01)	0.942(0.01)	0.938(0.01)
	dAiler	0.752(0.04)	0.684(0.02)	0.698(0.02)	0.680(0.02)	0.688(0.02)	0.693(0.02)	0.687(0.02)	0.705(0.02)	0.706(0.02)
	concreteStrength	0.933(0.22)	0.875(0.04)	0.883(0.05)	0.866(0.06)	0.944(0.04)	0.928(0.10)	0.925(0.05)	0.910(0.13)	0.906(0.13)
	acceleration	0.933(0.02)	0.865(0.03)	0.882(0.04)	0.865(0.03)	0.925(0.03)	0.926(0.02)	0.920(0.03)	0.934(0.03)	0.927(0.03)
	airfoild	0.244(0.18)	0.214(0.09)	0.228(0.09)	0.237(0.13)	0.201(0.10)	0.220(0.09)	0.198(0.10)	0.206(0.12)	0.183(0.12)
	servo	0.369(0.11)	0.538(0.07)	0.572(0.08)	0.533(0.08)	0.572(0.07)	0.584(0.09)	0.572(0.08)	0.577(0.09)	0.562(0.07)
	a6	0.258(0.23)	0.492(0.08)	0.505(0.10)	0.498(0.08)	0.494(0.08)	0.521(0.09)	0.523(0.09)	0.485(0.09)	0.492(0.10)
	Abalone	0.767(0.04)	0.708(0.02)	0.722(0.02)	0.703(0.02)	0.688(0.02)	0.683(0.01)	0.683(0.02)	0.697(0.02)	0.698(0.02)
	machineCpu	0.770(0.11)	0.763(0.08)	0.761(0.09)	0.757(0.09)	0.758(0.09)	0.768(0.09)	0.762(0.08)	0.768(0.09)	0.756(0.10)
	a3	0.187(0.11)	0.452(0.08)	0.464(0.08)	0.455(0.07)	0.478(0.08)	0.482(0.08)	0.481(0.07)	0.489(0.09)	0.476(0.08)
	a4	0.256(0.26)	0.518(0.09)	0.513(0.09)	0.491(0.10)	0.551(0.12)	0.529(0.14)	0.533(0.10)	0.541(0.15)	0.551(0.11)
	a1	0.103(0.16)	0.620(0.08)	0.606(0.09)	0.622(0.07)	0.660(0.11)	0.652(0.10)	0.670(0.09)	0.651(0.12)	0.666(0.11)
	a7	0.141(0.13)	0.349(0.05)	0.354(0.06)	0.335(0.07)	0.347(0.06)	0.348(0.05)	0.345(0.06)	0.340(0.06)	0.354(0.07)
SVM	boston	0.895(0.04)	0.855(0.04)	0.857(0.05)	0.853(0.05)	0.881(0.04)	0.882(0.04)	0.878(0.04)	0.886(0.04)	0.882(0.04)
	a2	0.282(0.38)	0.492(0.12)	0.481(0.13)	0.502(0.15)	0.501(0.15)	0.530(0.15)	0.516(0.15)	0.474(0.20)	0.469(0.19)
	a5	0.145(0.16)	0.513(0.09)	0.514(0.09)	0.520(0.12)	0.552(0.11)	0.572(0.10)	0.551(0.10)	0.541(0.13)	0.499(0.16)
	fuelCons	0.908(0.04)	0.844(0.04)	0.853(0.05)	0.838(0.04)	0.869(0.03)	0.864(0.04)	0.861(0.04)	0.886(0.03)	0.877(0.03)
	availPwr	0.934(0.02)	0.906(0.04)	0.914(0.03)	0.906(0.03)	0.930(0.01)	0.908(0.02)	0.908(0.02)	0.931(0.02)	0.930(0.01)
	cpuSm	0.163(0.03)	0.178(0.03)	0.181(0.03)	0.180(0.02)	0.180(0.03)	0.173(0.03)	0.170(0.03)	0.191(0.03)	0.189(0.03)
	maxTorque	0.973(0.01)	0.942(0.03)	0.956(0.02)	0.947(0.03)	0.969(0.01)	0.968(0.01)	0.966(0.01)	0.968(0.01)	0.967(0.01)
	bank8FM	0.949(0.01)	0.952(0.01)	0.952(0.01)	0.952(0.01)	0.951(0.01)	0.950(0.01)	0.949(0.01)	0.954(0.01)	0.954(0.01)
	dAiler	0.781(0.05)	0.711(0.02)	0.718(0.02)	0.707(0.02)	0.677(0.02)	0.669(0.01)	0.670(0.01)	0.703(0.02)	0.702(0.02)

Continued on next page

Table A.9 – continued from previous page

Learner	Data sets	None	U._._	U.F._	U.S._	S._._	S.F.F	S.S.F	S.F.S	S.S.S
SVM	concreteStrength	0.855(0.15)	0.827(0.07)	0.848(0.06)	0.830(0.06)	0.877(0.05)	0.878(0.05)	0.877(0.05)	0.885(0.05)	0.886(0.05)
	acceleration	0.876(0.08)	0.864(0.06)	0.872(0.06)	0.859(0.06)	0.869(0.04)	0.860(0.04)	0.860(0.04)	0.886(0.03)	0.885(0.03)
	airfoild	0.166(0.11)	0.222(0.12)	0.214(0.10)	0.231(0.11)	0.247(0.11)	0.225(0.10)	0.234(0.11)	0.248(0.13)	0.254(0.13)
NNET	servo	0.644(0.14)	0.587(0.16)	0.638(0.17)	0.608(0.17)	0.648(0.18)	0.644(0.18)	0.681(0.19)	0.643(0.18)	0.667(0.17)
	a6	0.175(0.19)	0.470(0.07)	0.470(0.06)	0.455(0.08)	0.473(0.06)	0.469(0.05)	0.469(0.06)	0.462(0.06)	0.465(0.07)
	Abalone	0.707(0.06)	0.656(0.08)	0.668(0.07)	0.655(0.06)	0.650(0.08)	0.651(0.06)	0.652(0.06)	0.662(0.07)	0.663(0.07)
	machineCpu	0.059(0.07)	0.454(0.07)	0.452(0.05)	0.462(0.06)	0.484(0.07)	0.361(0.24)	0.276(0.24)	0.479(0.07)	0.468(0.05)
	a3	0.203(0.22)	0.434(0.05)	0.439(0.05)	0.427(0.05)	0.453(0.06)	0.444(0.05)	0.441(0.06)	0.452(0.06)	0.443(0.08)
	a4	0.407(0.25)	0.427(0.13)	0.409(0.15)	0.425(0.12)	0.479(0.13)	0.481(0.12)	0.486(0.13)	0.497(0.12)	0.501(0.13)
	a1	0.252(0.27)	0.400(0.24)	0.439(0.18)	0.418(0.20)	0.386(0.25)	0.319(0.23)	0.338(0.23)	0.363(0.26)	0.371(0.26)
	a7	0.282(0.22)	0.326(0.05)	0.322(0.06)	0.338(0.06)	0.334(0.06)	0.332(0.06)	0.338(0.06)	0.336(0.06)	0.338(0.06)
	boston	0.308(0.28)	0.324(0.20)	0.343(0.20)	0.335(0.20)	0.276(0.22)	0.324(0.22)	0.318(0.24)	0.312(0.25)	0.324(0.23)
	a2	0.048(0.14)	0.325(0.15)	0.342(0.19)	0.361(0.14)	0.369(0.19)	0.355(0.20)	0.352(0.22)	0.373(0.21)	0.338(0.21)
	a5	0.011(0.04)	0.363(0.15)	0.355(0.15)	0.354(0.15)	0.362(0.17)	0.372(0.18)	0.359(0.19)	0.382(0.16)	0.378(0.18)
	fuelCons	0.065(0.15)	0.368(0.22)	0.343(0.24)	0.337(0.24)	0.290(0.24)	0.246(0.23)	0.262(0.25)	0.324(0.19)	0.325(0.22)
	availPwr	0.094(0.12)	0.166(0.20)	0.142(0.16)	0.147(0.16)	0.056(0.13)	0.049(0.13)	0.043(0.12)	0.038(0.11)	0.035(0.11)
	cpuSm	0.094(0.10)	0.317(0.24)	0.284(0.23)	0.304(0.23)	0.316(0.25)	0.267(0.29)	0.271(0.27)	0.253(0.24)	0.282(0.24)
	maxTorque	0.093(0.18)	0.397(0.21)	0.366(0.24)	0.390(0.21)	0.237(0.16)	0.209(0.24)	0.157(0.18)	0.270(0.25)	0.278(0.23)
	bank8FM	0.954(0.01)	0.890(0.10)	0.891(0.10)	0.891(0.10)	0.938(0.03)	0.912(0.06)	0.919(0.06)	0.918(0.10)	0.929(0.07)
	dAiler	0.351(0.09)	0.301(0.04)	0.303(0.04)	0.312(0.07)	0.334(0.06)	0.349(0.10)	0.352(0.11)	0.364(0.12)	0.349(0.10)
	concreteStrength	0.000(0.00)	0.258(0.20)	0.217(0.18)	0.258(0.17)	0.115(0.18)	0.087(0.17)	0.120(0.18)	0.190(0.22)	0.191(0.23)
	acceleration	0.116(0.29)	0.268(0.20)	0.258(0.20)	0.323(0.21)	0.235(0.23)	0.259(0.27)	0.254(0.27)	0.337(0.23)	0.335(0.23)
	airfoild	0.020(0.03)	0.078(0.07)	0.082(0.06)	0.079(0.06)	0.029(0.04)	0.034(0.05)	0.029(0.04)	0.046(0.06)	0.040(0.05)

Table A.10: Evaluation results concerning the rec^{ϕ} metric (average and standard deviation) for 20 data sets and all tested variants of 4 learners using biased pre-processing strategies.

Learner	Data sets	None	U._._	U.F._	U.S._	S._._	S.F.F	S.S.F	S.F.S	S.S.S
MARS	servo	0.683(0.14)	0.694(0.15)	0.690(0.15)	0.698(0.15)	0.718(0.14)	0.693(0.15)	0.706(0.14)	0.694(0.15)	0.700(0.14)
	a6	0.517(0.13)	0.573(0.16)	0.590(0.16)	0.574(0.16)	0.583(0.15)	0.594(0.14)	0.583(0.15)	0.577(0.16)	0.569(0.14)
	Abalone	0.687(0.02)	0.780(0.02)	0.769(0.02)	0.782(0.02)	0.795(0.02)	0.799(0.02)	0.798(0.02)	0.787(0.02)	0.786(0.02)
	machineCpu	0.806(0.09)	0.816(0.09)	0.787(0.11)	0.815(0.10)	0.809(0.10)	0.823(0.10)	0.827(0.09)	0.812(0.08)	0.811(0.08)
	a3	0.522(0.13)	0.645(0.13)	0.627(0.13)	0.640(0.16)	0.616(0.13)	0.631(0.14)	0.625(0.13)	0.619(0.14)	0.621(0.12)
	a4	0.568(0.13)	0.669(0.17)	0.624(0.18)	0.650(0.16)	0.638(0.15)	0.648(0.18)	0.669(0.15)	0.643(0.18)	0.671(0.14)
	a1	0.704(0.12)	0.795(0.12)	0.789(0.10)	0.794(0.12)	0.783(0.12)	0.790(0.12)	0.799(0.13)	0.779(0.12)	0.791(0.12)
	a7	0.360(0.18)	0.457(0.22)	0.431(0.22)	0.423(0.23)	0.440(0.22)	0.424(0.20)	0.401(0.22)	0.425(0.25)	0.430(0.21)
	boston	0.892(0.06)	0.905(0.05)	0.905(0.05)	0.907(0.05)	0.907(0.05)	0.905(0.05)	0.908(0.05)	0.897(0.06)	0.900(0.05)
	a2	0.564(0.24)	0.655(0.26)	0.646(0.25)	0.636(0.28)	0.623(0.26)	0.643(0.27)	0.645(0.28)	0.623(0.26)	0.631(0.27)
	a5	0.521(0.23)	0.637(0.26)	0.658(0.27)	0.651(0.27)	0.624(0.27)	0.609(0.27)	0.633(0.26)	0.592(0.27)	0.616(0.26)
	fuelCons	0.857(0.02)	0.898(0.02)	0.899(0.02)	0.902(0.02)	0.902(0.03)	0.905(0.03)	0.895(0.03)	0.903(0.02)	0.899(0.02)
	availPwr	0.904(0.02)	0.916(0.02)	0.918(0.02)	0.918(0.02)	0.916(0.02)	0.903(0.02)	0.900(0.02)	0.916(0.01)	0.915(0.01)
	cpuSm	0.141(0.03)	0.166(0.03)	0.170(0.03)	0.167(0.03)	0.169(0.03)	0.150(0.03)	0.150(0.03)	0.177(0.03)	0.177(0.03)
	maxTorque	0.955(0.01)	0.985(0.01)	0.983(0.01)	0.985(0.01)	0.966(0.02)	0.963(0.02)	0.964(0.02)	0.965(0.02)	0.964(0.02)
	bank8FM	0.938(0.01)	0.947(0.01)	0.947(0.01)	0.947(0.01)	0.947(0.01)	0.948(0.01)	0.948(0.01)	0.945(0.01)	0.945(0.01)
	dAiler	0.692(0.02)	0.822(0.02)	0.815(0.02)	0.824(0.02)	0.837(0.02)	0.841(0.02)	0.840(0.02)	0.821(0.02)	0.821(0.02)
	concreteStrength	0.880(0.04)	0.931(0.04)	0.934(0.03)	0.937(0.03)	0.932(0.03)	0.934(0.03)	0.934(0.03)	0.929(0.03)	0.928(0.03)
	acceleration	0.899(0.03)	0.937(0.02)	0.934(0.02)	0.936(0.02)	0.934(0.02)	0.933(0.02)	0.932(0.02)	0.929(0.02)	0.928(0.02)
	airfoild	0.155(0.07)	0.206(0.09)	0.197(0.11)	0.208(0.11)	0.216(0.11)	0.231(0.12)	0.211(0.11)	0.209(0.12)	0.201(0.11)
RF	servo	0.773(0.14)	0.777(0.14)	0.768(0.14)	0.772(0.14)	0.784(0.17)	0.770(0.16)	0.788(0.15)	0.779(0.16)	0.786(0.16)
	a6	0.523(0.17)	0.621(0.21)	0.620(0.19)	0.614(0.19)	0.586(0.21)	0.568(0.18)	0.583(0.17)	0.575(0.20)	0.589(0.18)
	Abalone	0.704(0.02)	0.787(0.02)	0.776(0.02)	0.788(0.02)	0.777(0.02)	0.784(0.02)	0.782(0.02)	0.770(0.02)	0.768(0.02)
	machineCpu	0.815(0.09)	0.821(0.09)	0.816(0.09)	0.820(0.08)	0.818(0.09)	0.804(0.08)	0.803(0.08)	0.809(0.10)	0.814(0.09)
	a3	0.528(0.12)	0.667(0.15)	0.643(0.12)	0.654(0.14)	0.644(0.13)	0.627(0.13)	0.645(0.13)	0.615(0.14)	0.630(0.14)
	a4	0.595(0.13)	0.690(0.17)	0.692(0.14)	0.702(0.16)	0.650(0.14)	0.663(0.14)	0.667(0.14)	0.657(0.14)	0.663(0.13)
	a1	0.721(0.12)	0.806(0.11)	0.796(0.12)	0.820(0.12)	0.771(0.12)	0.768(0.13)	0.788(0.12)	0.760(0.13)	0.778(0.13)

Continued on next page

Table A.10 – continued from previous page

Learner	Data sets	None	U._._	U.F._	U.S._	S._._	S.F.F	S.S.F	S.F.S	S.S.S
	a7	0.368(0.17)	0.477(0.26)	0.452(0.26)	0.461(0.26)	0.489(0.23)	0.427(0.21)	0.444(0.21)	0.428(0.22)	0.451(0.22)
	boston	0.898(0.06)	0.912(0.05)	0.910(0.05)	0.913(0.05)	0.909(0.06)	0.913(0.06)	0.914(0.06)	0.909(0.06)	0.909(0.06)
	a2	0.554(0.24)	0.686(0.27)	0.660(0.26)	0.679(0.27)	0.631(0.25)	0.652(0.27)	0.652(0.26)	0.629(0.25)	0.629(0.25)
	a5	0.541(0.23)	0.672(0.27)	0.680(0.27)	0.676(0.28)	0.617(0.26)	0.618(0.27)	0.626(0.27)	0.598(0.26)	0.604(0.26)
	fuelCons	0.908(0.02)	0.928(0.02)	0.927(0.02)	0.928(0.02)	0.938(0.02)	0.932(0.02)	0.935(0.02)	0.932(0.02)	0.935(0.02)
	availPwr	0.965(0.02)	0.966(0.01)	0.966(0.01)	0.966(0.01)	0.971(0.01)	0.969(0.02)	0.969(0.02)	0.970(0.02)	0.969(0.02)
RF	cpuSm	0.492(0.05)	0.518(0.07)	0.512(0.06)	0.513(0.06)	0.511(0.06)	0.506(0.06)	0.506(0.06)	0.511(0.06)	0.514(0.06)
	maxTorque	0.966(0.01)	0.971(0.01)	0.971(0.01)	0.971(0.01)	0.975(0.01)	0.974(0.01)	0.975(0.01)	0.973(0.01)	0.973(0.01)
	bank8FM	0.945(0.01)	0.957(0.01)	0.956(0.01)	0.958(0.01)	0.957(0.01)	0.956(0.01)	0.957(0.01)	0.955(0.01)	0.956(0.01)
	dAiler	0.720(0.02)	0.837(0.02)	0.827(0.02)	0.843(0.02)	0.814(0.02)	0.817(0.02)	0.815(0.02)	0.798(0.02)	0.796(0.02)
	concreteStrength	0.927(0.03)	0.943(0.03)	0.941(0.03)	0.943(0.03)	0.941(0.03)	0.944(0.03)	0.949(0.03)	0.937(0.03)	0.941(0.03)
	acceleration	0.935(0.02)	0.951(0.02)	0.948(0.02)	0.950(0.02)	0.958(0.02)	0.951(0.02)	0.951(0.02)	0.953(0.02)	0.954(0.02)
	airfoild	0.210(0.13)	0.219(0.10)	0.255(0.11)	0.227(0.12)	0.200(0.09)	0.229(0.09)	0.200(0.10)	0.189(0.09)	0.169(0.09)
	servo	0.421(0.14)	0.717(0.14)	0.723(0.13)	0.735(0.14)	0.754(0.14)	0.746(0.15)	0.753(0.15)	0.732(0.15)	0.734(0.14)
	a6	0.462(0.11)	0.582(0.17)	0.577(0.15)	0.587(0.14)	0.568(0.14)	0.567(0.14)	0.583(0.14)	0.541(0.13)	0.569(0.14)
	Abalone	0.665(0.02)	0.764(0.02)	0.748(0.02)	0.766(0.02)	0.786(0.02)	0.791(0.02)	0.790(0.02)	0.774(0.02)	0.773(0.02)
	machineCpu	0.798(0.10)	0.814(0.09)	0.804(0.09)	0.813(0.09)	0.811(0.09)	0.820(0.09)	0.823(0.09)	0.813(0.09)	0.807(0.10)
	a3	0.447(0.08)	0.631(0.15)	0.614(0.14)	0.619(0.15)	0.610(0.15)	0.635(0.14)	0.624(0.14)	0.605(0.15)	0.590(0.14)
	a4	0.519(0.11)	0.649(0.16)	0.628(0.16)	0.644(0.14)	0.627(0.16)	0.669(0.14)	0.643(0.15)	0.640(0.14)	0.627(0.16)
	a1	0.608(0.08)	0.789(0.11)	0.779(0.10)	0.806(0.10)	0.792(0.11)	0.802(0.11)	0.800(0.11)	0.780(0.12)	0.782(0.12)
	a7	0.345(0.13)	0.373(0.23)	0.383(0.21)	0.361(0.23)	0.398(0.22)	0.409(0.19)	0.409(0.21)	0.374(0.21)	0.380(0.23)
SVM	boston	0.873(0.05)	0.904(0.05)	0.903(0.05)	0.906(0.04)	0.913(0.05)	0.919(0.04)	0.919(0.04)	0.906(0.05)	0.907(0.05)
	a2	0.529(0.24)	0.619(0.26)	0.591(0.26)	0.606(0.27)	0.625(0.26)	0.608(0.27)	0.613(0.26)	0.578(0.25)	0.575(0.24)
	a5	0.487(0.20)	0.645(0.28)	0.637(0.26)	0.645(0.28)	0.624(0.27)	0.620(0.27)	0.624(0.27)	0.615(0.27)	0.613(0.27)
	fuelCons	0.908(0.02)	0.920(0.02)	0.920(0.02)	0.919(0.02)	0.918(0.03)	0.916(0.02)	0.917(0.03)	0.913(0.03)	0.916(0.03)
	availPwr	0.937(0.01)	0.943(0.01)	0.944(0.01)	0.944(0.01)	0.945(0.01)	0.936(0.02)	0.936(0.02)	0.945(0.01)	0.945(0.01)
	cpuSm	0.159(0.03)	0.160(0.04)	0.162(0.03)	0.162(0.03)	0.179(0.03)	0.178(0.04)	0.174(0.04)	0.181(0.03)	0.179(0.03)
	maxTorque	0.973(0.01)	0.975(0.01)	0.975(0.01)	0.975(0.01)	0.975(0.01)	0.973(0.01)	0.973(0.01)	0.974(0.01)	0.974(0.01)
	bank8FM	0.945(0.01)	0.945(0.01)	0.945(0.01)	0.945(0.01)	0.948(0.01)	0.949(0.01)	0.949(0.01)	0.946(0.01)	0.946(0.01)
	dAiler	0.684(0.02)	0.820(0.02)	0.813(0.02)	0.821(0.02)	0.843(0.02)	0.847(0.02)	0.847(0.02)	0.824(0.02)	0.823(0.02)

Continued on next page

Table A.10 – continued from previous page

Learner	Data sets	None	U._._	U.F._	U.S._	S._._	S.F.F	S.S.F	S.F.S	S.S.S
SVM	concreteStrength	0.874(0.04)	0.934(0.03)	0.934(0.03)	0.937(0.03)	0.945(0.03)	0.945(0.03)	0.945(0.03)	0.940(0.03)	0.939(0.03)
	acceleration	0.883(0.03)	0.937(0.02)	0.933(0.03)	0.936(0.02)	0.933(0.02)	0.935(0.02)	0.935(0.02)	0.927(0.02)	0.927(0.02)
	airfoild	0.162(0.08)	0.218(0.12)	0.212(0.11)	0.218(0.11)	0.238(0.11)	0.217(0.10)	0.222(0.10)	0.246(0.12)	0.248(0.12)
NNET	servo	0.699(0.15)	0.666(0.19)	0.709(0.19)	0.690(0.20)	0.694(0.19)	0.694(0.20)	0.720(0.20)	0.697(0.18)	0.713(0.18)
	a6	0.479(0.14)	0.593(0.20)	0.606(0.19)	0.605(0.19)	0.594(0.19)	0.577(0.18)	0.580(0.18)	0.572(0.18)	0.585(0.18)
	Abalone	0.690(0.04)	0.774(0.05)	0.763(0.04)	0.777(0.04)	0.790(0.06)	0.803(0.05)	0.803(0.05)	0.779(0.05)	0.779(0.05)
	machineCpu	0.496(0.12)	0.671(0.18)	0.687(0.18)	0.674(0.18)	0.676(0.18)	0.640(0.17)	0.626(0.18)	0.683(0.17)	0.665(0.17)
	a3	0.473(0.10)	0.627(0.16)	0.618(0.15)	0.610(0.14)	0.606(0.13)	0.616(0.14)	0.603(0.14)	0.605(0.14)	0.598(0.14)
	a4	0.567(0.14)	0.668(0.16)	0.667(0.16)	0.651(0.16)	0.674(0.15)	0.669(0.15)	0.679(0.15)	0.680(0.15)	0.680(0.16)
	a1	0.629(0.13)	0.747(0.11)	0.741(0.11)	0.745(0.12)	0.752(0.11)	0.740(0.12)	0.744(0.10)	0.747(0.12)	0.741(0.11)
	a7	0.395(0.21)	0.434(0.22)	0.417(0.21)	0.450(0.22)	0.446(0.22)	0.421(0.23)	0.453(0.24)	0.449(0.23)	0.444(0.23)
	boston	0.688(0.09)	0.790(0.06)	0.800(0.06)	0.786(0.06)	0.774(0.06)	0.783(0.06)	0.777(0.07)	0.780(0.07)	0.781(0.07)
	a2	0.505(0.21)	0.650(0.27)	0.636(0.26)	0.660(0.27)	0.641(0.27)	0.651(0.27)	0.654(0.27)	0.633(0.26)	0.635(0.25)
	a5	0.466(0.19)	0.644(0.27)	0.639(0.27)	0.644(0.27)	0.631(0.27)	0.626(0.27)	0.631(0.27)	0.636(0.27)	0.634(0.27)
	fuelCons	0.445(0.05)	0.573(0.06)	0.590(0.06)	0.567(0.07)	0.555(0.10)	0.550(0.09)	0.551(0.10)	0.566(0.09)	0.573(0.09)
	availPwr	0.558(0.06)	0.740(0.06)	0.735(0.06)	0.731(0.06)	0.715(0.06)	0.697(0.05)	0.695(0.05)	0.718(0.06)	0.714(0.05)
	cpuSm	0.164(0.06)	0.217(0.13)	0.193(0.13)	0.214(0.12)	0.198(0.13)	0.198(0.11)	0.209(0.12)	0.181(0.14)	0.201(0.14)
	maxTorque	0.593(0.05)	0.774(0.05)	0.779(0.05)	0.766(0.05)	0.759(0.05)	0.759(0.05)	0.747(0.05)	0.767(0.05)	0.759(0.05)
	bank8FM	0.953(0.01)	0.947(0.03)	0.947(0.03)	0.947(0.03)	0.958(0.01)	0.951(0.02)	0.953(0.02)	0.953(0.03)	0.955(0.02)
	dAiler	0.535(0.02)	0.594(0.04)	0.595(0.04)	0.595(0.04)	0.636(0.03)	0.635(0.05)	0.634(0.05)	0.637(0.04)	0.633(0.04)
	concreteStrength	0.632(0.04)	0.827(0.05)	0.834(0.04)	0.827(0.05)	0.796(0.05)	0.793(0.05)	0.795(0.05)	0.817(0.05)	0.810(0.05)
	acceleration	0.629(0.05)	0.801(0.04)	0.811(0.04)	0.808(0.04)	0.796(0.04)	0.782(0.05)	0.785(0.05)	0.807(0.05)	0.804(0.05)
	airfoild	0.254(0.04)	0.170(0.04)	0.161(0.05)	0.174(0.05)	0.177(0.03)	0.174(0.04)	0.179(0.03)	0.169(0.04)	0.172(0.04)

Table A.11: Evaluation results concerning the $spec^{\phi}$ metric (average and standard deviation) for 20 data sets and all tested variants of 4 learners using biased pre-processing strategies.

Learner	Data sets	None	U._._	U.F._	U.S._	S._._	S.F.F	S.S.F	S.F.S	S.S.S
MARS	servo	0.792(0.09)	0.788(0.09)	0.786(0.09)	0.790(0.09)	0.805(0.09)	0.787(0.09)	0.797(0.09)	0.789(0.09)	0.796(0.08)
	a6	0.690(0.09)	0.696(0.10)	0.706(0.09)	0.694(0.11)	0.710(0.09)	0.722(0.08)	0.711(0.09)	0.708(0.10)	0.703(0.10)
	Abalone	0.784(0.01)	0.828(0.01)	0.824(0.01)	0.829(0.01)	0.832(0.01)	0.831(0.01)	0.831(0.01)	0.830(0.01)	0.830(0.01)
	machineCpu	0.887(0.05)	0.888(0.05)	0.873(0.06)	0.887(0.05)	0.887(0.05)	0.894(0.05)	0.896(0.05)	0.889(0.04)	0.888(0.04)
	a3	0.695(0.10)	0.716(0.08)	0.707(0.08)	0.706(0.10)	0.716(0.08)	0.717(0.08)	0.719(0.08)	0.717(0.09)	0.719(0.08)
	a4	0.720(0.08)	0.756(0.10)	0.728(0.11)	0.748(0.09)	0.746(0.09)	0.752(0.11)	0.765(0.09)	0.750(0.12)	0.769(0.08)
	a1	0.816(0.07)	0.856(0.07)	0.854(0.06)	0.855(0.07)	0.851(0.07)	0.856(0.07)	0.860(0.07)	0.851(0.07)	0.856(0.07)
	a7	0.544(0.18)	0.547(0.18)	0.527(0.19)	0.513(0.21)	0.540(0.19)	0.540(0.17)	0.515(0.19)	0.525(0.21)	0.540(0.17)
	boston	0.937(0.03)	0.941(0.03)	0.942(0.03)	0.942(0.03)	0.943(0.03)	0.942(0.03)	0.944(0.03)	0.938(0.03)	0.940(0.03)
	a2	0.691(0.25)	0.707(0.25)	0.707(0.25)	0.697(0.26)	0.702(0.25)	0.715(0.26)	0.713(0.26)	0.710(0.26)	0.710(0.26)
	a5	0.664(0.24)	0.701(0.25)	0.708(0.26)	0.705(0.26)	0.706(0.26)	0.700(0.26)	0.712(0.26)	0.688(0.25)	0.702(0.25)
	fuelCons	0.900(0.01)	0.923(0.01)	0.923(0.01)	0.926(0.01)	0.924(0.01)	0.925(0.01)	0.919(0.01)	0.925(0.01)	0.923(0.01)
	availPwr	0.946(0.01)	0.952(0.01)	0.953(0.01)	0.952(0.01)	0.951(0.01)	0.944(0.01)	0.942(0.01)	0.951(0.01)	0.951(0.01)
	cpuSm	0.345(0.04)	0.369(0.03)	0.373(0.03)	0.370(0.03)	0.373(0.03)	0.348(0.04)	0.348(0.04)	0.382(0.03)	0.382(0.03)
	maxTorque	0.973(0.01)	0.991(0.00)	0.990(0.01)	0.991(0.00)	0.980(0.01)	0.979(0.01)	0.979(0.01)	0.979(0.01)	0.979(0.01)
	bank8FM	0.961(0.00)	0.967(0.00)	0.966(0.00)	0.967(0.00)	0.966(0.00)	0.967(0.00)	0.967(0.00)	0.965(0.00)	0.965(0.00)
	dAiler	0.807(0.01)	0.877(0.01)	0.874(0.01)	0.878(0.01)	0.881(0.01)	0.883(0.01)	0.882(0.01)	0.875(0.01)	0.875(0.01)
	concreteStrength	0.929(0.02)	0.954(0.02)	0.955(0.02)	0.957(0.01)	0.956(0.02)	0.957(0.02)	0.957(0.02)	0.954(0.02)	0.954(0.02)
	acceleration	0.941(0.01)	0.960(0.01)	0.958(0.01)	0.959(0.01)	0.959(0.01)	0.959(0.01)	0.958(0.01)	0.955(0.01)	0.955(0.01)
	airfoild	0.346(0.08)	0.394(0.10)	0.382(0.11)	0.392(0.12)	0.407(0.11)	0.418(0.12)	0.400(0.11)	0.398(0.11)	0.389(0.12)
RF	servo	0.864(0.09)	0.853(0.08)	0.850(0.09)	0.848(0.09)	0.865(0.10)	0.854(0.10)	0.869(0.09)	0.860(0.10)	0.866(0.10)
	a6	0.692(0.12)	0.717(0.12)	0.721(0.12)	0.715(0.11)	0.715(0.13)	0.708(0.12)	0.716(0.10)	0.708(0.13)	0.717(0.11)
	Abalone	0.794(0.01)	0.830(0.01)	0.827(0.01)	0.830(0.01)	0.827(0.01)	0.829(0.01)	0.828(0.01)	0.824(0.01)	0.823(0.01)
	machineCpu	0.893(0.05)	0.893(0.05)	0.892(0.05)	0.894(0.04)	0.894(0.05)	0.887(0.04)	0.886(0.04)	0.889(0.05)	0.892(0.05)
	a3	0.698(0.09)	0.723(0.09)	0.718(0.07)	0.713(0.09)	0.744(0.08)	0.731(0.08)	0.741(0.08)	0.731(0.09)	0.739(0.08)
	a4	0.738(0.08)	0.773(0.10)	0.776(0.08)	0.781(0.09)	0.762(0.08)	0.771(0.08)	0.773(0.08)	0.769(0.08)	0.771(0.08)
	a1	0.828(0.07)	0.864(0.06)	0.861(0.07)	0.871(0.06)	0.850(0.07)	0.849(0.07)	0.859(0.07)	0.844(0.07)	0.853(0.07)

Continued on next page

Table A.11 – continued from previous page

Learner	Data sets	None	U._._	U.F._	U.S._	S._._	S.F.F	S.S.F	S.F.S	S.S.S
	a7	0.555(0.18)	0.542(0.19)	0.523(0.20)	0.532(0.20)	0.592(0.19)	0.548(0.19)	0.561(0.18)	0.548(0.20)	0.564(0.19)
	boston	0.942(0.03)	0.948(0.03)	0.947(0.03)	0.947(0.03)	0.947(0.03)	0.948(0.03)	0.949(0.03)	0.947(0.03)	0.947(0.03)
	a2	0.686(0.25)	0.738(0.26)	0.723(0.26)	0.734(0.26)	0.724(0.26)	0.736(0.27)	0.736(0.26)	0.725(0.26)	0.725(0.26)
	a5	0.677(0.25)	0.715(0.26)	0.722(0.26)	0.725(0.26)	0.710(0.26)	0.713(0.26)	0.714(0.26)	0.702(0.26)	0.704(0.26)
	fuelCons	0.937(0.01)	0.935(0.01)	0.936(0.01)	0.935(0.01)	0.947(0.01)	0.943(0.01)	0.944(0.01)	0.944(0.01)	0.946(0.01)
	availPwr	0.981(0.01)	0.977(0.01)	0.978(0.01)	0.977(0.01)	0.983(0.01)	0.983(0.01)	0.983(0.01)	0.983(0.01)	0.983(0.01)
RF	cpuSm	0.651(0.04)	0.660(0.04)	0.657(0.04)	0.657(0.04)	0.661(0.04)	0.656(0.04)	0.655(0.04)	0.662(0.04)	0.663(0.04)
	maxTorque	0.982(0.01)	0.980(0.01)	0.980(0.01)	0.979(0.01)	0.985(0.00)	0.985(0.01)	0.985(0.01)	0.985(0.01)	0.985(0.01)
	bank8FM	0.966(0.00)	0.971(0.00)	0.971(0.00)	0.971(0.00)	0.972(0.00)	0.971(0.00)	0.972(0.00)	0.971(0.00)	0.971(0.00)
	dAiler	0.825(0.01)	0.885(0.01)	0.881(0.01)	0.887(0.01)	0.874(0.01)	0.876(0.01)	0.874(0.01)	0.867(0.01)	0.865(0.01)
	concreteStrength	0.957(0.02)	0.959(0.02)	0.958(0.02)	0.958(0.01)	0.963(0.02)	0.964(0.02)	0.966(0.01)	0.961(0.02)	0.962(0.02)
	acceleration	0.962(0.01)	0.967(0.01)	0.966(0.01)	0.967(0.01)	0.973(0.01)	0.970(0.01)	0.970(0.01)	0.971(0.01)	0.971(0.01)
	airfoild	0.400(0.14)	0.409(0.10)	0.437(0.11)	0.412(0.12)	0.391(0.11)	0.425(0.10)	0.389(0.12)	0.382(0.11)	0.359(0.11)
	servo	0.601(0.13)	0.774(0.09)	0.793(0.08)	0.778(0.09)	0.804(0.08)	0.806(0.09)	0.804(0.09)	0.798(0.09)	0.793(0.09)
	a6	0.657(0.08)	0.709(0.11)	0.713(0.09)	0.716(0.08)	0.711(0.09)	0.714(0.09)	0.721(0.09)	0.698(0.08)	0.711(0.09)
	Abalone	0.773(0.01)	0.825(0.01)	0.817(0.01)	0.825(0.01)	0.833(0.01)	0.834(0.01)	0.833(0.01)	0.828(0.01)	0.828(0.01)
	machineCpu	0.883(0.05)	0.890(0.05)	0.885(0.05)	0.889(0.05)	0.889(0.05)	0.894(0.05)	0.895(0.05)	0.890(0.05)	0.886(0.05)
	a3	0.644(0.06)	0.716(0.09)	0.714(0.08)	0.706(0.09)	0.720(0.09)	0.730(0.08)	0.725(0.08)	0.719(0.09)	0.711(0.09)
	a4	0.688(0.08)	0.749(0.09)	0.734(0.10)	0.746(0.08)	0.746(0.10)	0.770(0.08)	0.754(0.09)	0.757(0.09)	0.746(0.10)
	a1	0.759(0.05)	0.844(0.06)	0.835(0.06)	0.849(0.06)	0.853(0.06)	0.860(0.06)	0.858(0.06)	0.849(0.07)	0.850(0.07)
	a7	0.549(0.15)	0.487(0.21)	0.511(0.19)	0.484(0.20)	0.521(0.20)	0.537(0.18)	0.530(0.19)	0.511(0.20)	0.511(0.21)
SVM	boston	0.928(0.03)	0.942(0.03)	0.942(0.03)	0.943(0.02)	0.948(0.03)	0.951(0.02)	0.951(0.02)	0.945(0.03)	0.945(0.03)
	a2	0.664(0.25)	0.696(0.25)	0.682(0.25)	0.689(0.26)	0.711(0.26)	0.703(0.26)	0.704(0.26)	0.689(0.25)	0.685(0.25)
	a5	0.641(0.23)	0.707(0.26)	0.709(0.26)	0.714(0.26)	0.711(0.26)	0.713(0.26)	0.712(0.26)	0.706(0.26)	0.702(0.26)
	fuelCons	0.935(0.01)	0.933(0.01)	0.933(0.01)	0.931(0.01)	0.936(0.01)	0.935(0.01)	0.936(0.01)	0.934(0.01)	0.935(0.01)
	availPwr	0.964(0.01)	0.966(0.01)	0.966(0.01)	0.966(0.01)	0.968(0.01)	0.963(0.01)	0.963(0.01)	0.968(0.01)	0.968(0.01)
	cpuSm	0.367(0.04)	0.355(0.04)	0.358(0.04)	0.358(0.04)	0.384(0.03)	0.381(0.04)	0.377(0.04)	0.386(0.04)	0.384(0.04)
	maxTorque	0.984(0.00)	0.984(0.00)	0.985(0.00)	0.984(0.00)	0.984(0.00)	0.984(0.00)	0.984(0.01)	0.984(0.00)	0.984(0.00)
	bank8FM	0.965(0.00)	0.966(0.00)	0.966(0.00)	0.966(0.00)	0.968(0.00)	0.968(0.00)	0.968(0.00)	0.967(0.00)	0.966(0.00)
	dAiler	0.800(0.01)	0.877(0.01)	0.874(0.01)	0.878(0.01)	0.886(0.01)	0.887(0.01)	0.887(0.01)	0.879(0.01)	0.878(0.01)

Continued on next page

Table A.11 – continued from previous page

Learner	Data sets	None	U._._	U.F._	U.S._	S._._	S.F.F	S.S.F	S.F.S	S.S.S
SVM	concreteStrength	0.925(0.02)	0.953(0.02)	0.955(0.01)	0.955(0.02)	0.962(0.01)	0.962(0.01)	0.962(0.01)	0.960(0.01)	0.959(0.01)
	acceleration	0.933(0.01)	0.961(0.01)	0.959(0.01)	0.960(0.01)	0.959(0.01)	0.960(0.01)	0.960(0.01)	0.956(0.01)	0.956(0.01)
	airfoild	0.354(0.10)	0.403(0.12)	0.394(0.12)	0.406(0.11)	0.429(0.11)	0.410(0.10)	0.415(0.10)	0.435(0.12)	0.438(0.11)
NNET	servo	0.802(0.10)	0.756(0.14)	0.792(0.14)	0.771(0.14)	0.787(0.14)	0.786(0.14)	0.810(0.14)	0.786(0.14)	0.801(0.13)
	a6	0.665(0.10)	0.690(0.12)	0.699(0.11)	0.697(0.11)	0.698(0.12)	0.689(0.11)	0.693(0.11)	0.688(0.11)	0.693(0.11)
	Abalone	0.785(0.02)	0.822(0.03)	0.819(0.03)	0.824(0.03)	0.828(0.03)	0.833(0.03)	0.833(0.03)	0.825(0.03)	0.824(0.03)
	machineCpu	0.685(0.09)	0.711(0.10)	0.712(0.10)	0.716(0.10)	0.730(0.10)	0.740(0.10)	0.730(0.11)	0.730(0.09)	0.719(0.09)
	a3	0.662(0.07)	0.690(0.09)	0.687(0.09)	0.682(0.08)	0.695(0.08)	0.693(0.08)	0.691(0.08)	0.693(0.08)	0.692(0.08)
	a4	0.714(0.09)	0.740(0.10)	0.742(0.09)	0.734(0.09)	0.760(0.08)	0.754(0.09)	0.759(0.08)	0.763(0.09)	0.763(0.09)
	a1	0.767(0.08)	0.815(0.07)	0.813(0.06)	0.818(0.07)	0.824(0.06)	0.815(0.07)	0.819(0.06)	0.819(0.07)	0.816(0.06)
	a7	0.562(0.20)	0.520(0.19)	0.511(0.19)	0.533(0.19)	0.537(0.18)	0.518(0.20)	0.538(0.20)	0.539(0.19)	0.535(0.20)
	boston	0.814(0.05)	0.855(0.04)	0.858(0.03)	0.855(0.04)	0.848(0.04)	0.854(0.04)	0.852(0.04)	0.851(0.04)	0.853(0.04)
	a2	0.654(0.23)	0.703(0.25)	0.696(0.25)	0.710(0.25)	0.705(0.26)	0.711(0.26)	0.713(0.26)	0.709(0.25)	0.708(0.25)
	a5	0.628(0.23)	0.687(0.25)	0.685(0.25)	0.692(0.25)	0.691(0.25)	0.690(0.25)	0.693(0.25)	0.693(0.25)	0.692(0.25)
	fuelCons	0.602(0.04)	0.684(0.04)	0.694(0.04)	0.682(0.04)	0.672(0.06)	0.672(0.06)	0.671(0.06)	0.678(0.06)	0.684(0.05)
	availPwr	0.729(0.04)	0.815(0.04)	0.811(0.04)	0.810(0.04)	0.796(0.04)	0.793(0.03)	0.792(0.03)	0.795(0.03)	0.793(0.03)
	cpuSm	0.359(0.06)	0.364(0.13)	0.339(0.12)	0.364(0.12)	0.343(0.13)	0.355(0.10)	0.367(0.11)	0.329(0.13)	0.352(0.13)
	maxTorque	0.751(0.03)	0.846(0.03)	0.847(0.03)	0.842(0.03)	0.837(0.03)	0.839(0.03)	0.832(0.03)	0.840(0.03)	0.835(0.03)
	bank8FM	0.970(0.00)	0.965(0.02)	0.965(0.02)	0.965(0.02)	0.972(0.01)	0.968(0.01)	0.969(0.01)	0.969(0.02)	0.970(0.01)
	dAiler	0.690(0.01)	0.721(0.02)	0.723(0.02)	0.722(0.02)	0.747(0.02)	0.746(0.03)	0.744(0.03)	0.749(0.03)	0.746(0.03)
	concreteStrength	0.775(0.03)	0.886(0.03)	0.889(0.02)	0.886(0.03)	0.870(0.03)	0.868(0.03)	0.870(0.03)	0.880(0.03)	0.877(0.03)
	acceleration	0.777(0.03)	0.869(0.02)	0.873(0.02)	0.873(0.02)	0.868(0.02)	0.864(0.02)	0.866(0.03)	0.870(0.03)	0.869(0.02)
	airfoild	0.458(0.04)	0.342(0.05)	0.328(0.05)	0.345(0.05)	0.351(0.03)	0.348(0.04)	0.354(0.03)	0.339(0.05)	0.344(0.04)

Table A.12: Evaluation results concerning the $NPval^{\phi}$ metric (average and standard deviation) for 20 data sets and all tested variants of 4 learners using biased pre-processing strategies.

Learner	Data sets	None	U._._	U.F._	U.S._	S._._	S.F.F	S.S.F	S.F.S	S.S.S
MARS	servo	0.968(0.04)	0.977(0.06)	0.972(0.05)	0.978(0.06)	0.967(0.03)	0.974(0.03)	0.966(0.03)	0.971(0.04)	0.964(0.05)
	a6	0.842(0.04)	0.766(0.13)	0.799(0.08)	0.821(0.06)	0.817(0.06)	0.834(0.05)	0.825(0.05)	0.828(0.07)	0.822(0.07)
	Abalone	0.896(0.01)	0.822(0.01)	0.820(0.01)	0.823(0.01)	0.801(0.01)	0.782(0.01)	0.787(0.01)	0.815(0.01)	0.819(0.01)
	machineCpu	0.977(0.02)	0.949(0.03)	0.959(0.02)	0.949(0.03)	0.961(0.02)	0.957(0.03)	0.958(0.03)	0.965(0.02)	0.966(0.02)
	a3	0.836(0.05)	0.743(0.09)	0.651(0.22)	0.761(0.11)	0.788(0.06)	0.801(0.07)	0.803(0.07)	0.806(0.06)	0.804(0.06)
	a4	0.869(0.05)	0.817(0.07)	0.793(0.07)	0.796(0.06)	0.820(0.06)	0.816(0.05)	0.818(0.06)	0.824(0.06)	0.829(0.05)
	a1	0.934(0.04)	0.863(0.06)	0.882(0.05)	0.877(0.05)	0.890(0.04)	0.886(0.05)	0.887(0.05)	0.905(0.04)	0.912(0.04)
	a7	0.873(0.05)	0.741(0.18)	0.728(0.22)	0.731(0.22)	0.819(0.10)	0.831(0.09)	0.826(0.07)	0.836(0.07)	0.819(0.09)
	boston	0.992(0.02)	0.957(0.02)	0.960(0.02)	0.955(0.02)	0.968(0.02)	0.963(0.02)	0.962(0.02)	0.967(0.02)	0.972(0.02)
	a2	0.914(0.04)	0.828(0.10)	0.789(0.11)	0.834(0.09)	0.839(0.06)	0.851(0.06)	0.835(0.05)	0.846(0.05)	0.839(0.06)
	a5	0.901(0.04)	0.774(0.10)	0.760(0.14)	0.820(0.09)	0.839(0.07)	0.845(0.05)	0.855(0.06)	0.839(0.05)	0.825(0.06)
	fuelCons	0.954(0.01)	0.934(0.01)	0.929(0.01)	0.933(0.01)	0.932(0.01)	0.927(0.01)	0.927(0.01)	0.933(0.01)	0.938(0.01)
	availPwr	0.988(0.00)	0.968(0.01)	0.971(0.01)	0.971(0.01)	0.971(0.01)	0.960(0.01)	0.960(0.01)	0.974(0.01)	0.973(0.01)
	cpuSm	0.854(0.01)	0.841(0.01)	0.835(0.01)	0.844(0.01)	0.846(0.01)	0.815(0.01)	0.816(0.01)	0.851(0.01)	0.853(0.01)
	maxTorque	0.994(0.00)	0.988(0.00)	0.987(0.01)	0.988(0.00)	0.976(0.01)	0.974(0.01)	0.974(0.01)	0.978(0.01)	0.977(0.01)
	bank8FM	0.995(0.00)	0.985(0.00)	0.985(0.00)	0.985(0.00)	0.986(0.00)	0.985(0.00)	0.986(0.00)	0.988(0.00)	0.989(0.00)
	dAiler	0.942(0.00)	0.859(0.01)	0.862(0.01)	0.859(0.01)	0.828(0.01)	0.818(0.01)	0.821(0.01)	0.858(0.01)	0.859(0.01)
	concreteStrength	0.998(0.01)	0.962(0.02)	0.955(0.02)	0.958(0.02)	0.980(0.01)	0.978(0.01)	0.978(0.01)	0.983(0.01)	0.982(0.01)
	acceleration	0.990(0.01)	0.967(0.01)	0.971(0.01)	0.972(0.01)	0.971(0.01)	0.960(0.01)	0.960(0.01)	0.979(0.01)	0.979(0.01)
	airfoild	0.835(0.02)	0.792(0.03)	0.781(0.03)	0.795(0.02)	0.816(0.02)	0.813(0.02)	0.815(0.02)	0.813(0.02)	0.817(0.02)
RF	servo	0.988(0.02)	1.000(0.02)	0.998(0.02)	1.000(0.02)	0.988(0.03)	0.989(0.03)	0.982(0.03)	0.992(0.02)	0.992(0.03)
	a6	0.835(0.04)	0.742(0.05)	0.739(0.06)	0.740(0.06)	0.805(0.06)	0.794(0.06)	0.810(0.06)	0.803(0.06)	0.808(0.06)
	Abalone	0.903(0.01)	0.842(0.01)	0.833(0.01)	0.847(0.01)	0.855(0.01)	0.845(0.01)	0.851(0.01)	0.865(0.01)	0.869(0.01)
	machineCpu	0.982(0.02)	0.950(0.05)	0.954(0.03)	0.955(0.03)	0.977(0.02)	0.971(0.03)	0.979(0.02)	0.970(0.03)	0.975(0.02)
	a3	0.838(0.05)	0.708(0.07)	0.677(0.07)	0.693(0.06)	0.781(0.06)	0.807(0.07)	0.792(0.06)	0.809(0.05)	0.797(0.06)
	a4	0.875(0.04)	0.744(0.06)	0.737(0.05)	0.739(0.04)	0.812(0.04)	0.809(0.05)	0.807(0.04)	0.820(0.04)	0.822(0.04)
	a1	0.929(0.03)	0.844(0.05)	0.843(0.05)	0.853(0.06)	0.899(0.04)	0.893(0.04)	0.911(0.04)	0.913(0.04)	0.921(0.04)

Continued on next page

Table A.12 – continued from previous page

Learner	Data sets	None	U._._	U.F._	U.S._	S._._	S.F.F	S.S.F	S.F.S	S.S.S
	a7	0.856(0.06)	0.666(0.09)	0.641(0.08)	0.666(0.08)	0.756(0.08)	0.757(0.06)	0.767(0.08)	0.764(0.07)	0.761(0.06)
	boston	0.992(0.01)	0.957(0.02)	0.947(0.02)	0.959(0.03)	0.984(0.02)	0.981(0.02)	0.983(0.01)	0.980(0.02)	0.982(0.02)
	a2	0.912(0.04)	0.757(0.06)	0.758(0.07)	0.753(0.05)	0.864(0.05)	0.859(0.06)	0.849(0.06)	0.864(0.06)	0.847(0.06)
	a5	0.900(0.04)	0.679(0.06)	0.705(0.04)	0.730(0.07)	0.847(0.05)	0.860(0.05)	0.854(0.04)	0.854(0.04)	0.854(0.05)
	fuelCons	0.979(0.00)	0.946(0.01)	0.945(0.01)	0.945(0.01)	0.963(0.01)	0.963(0.01)	0.963(0.01)	0.965(0.01)	0.965(0.01)
	availPwr	0.999(0.00)	0.981(0.01)	0.981(0.01)	0.981(0.01)	0.995(0.00)	0.995(0.00)	0.995(0.00)	0.996(0.00)	0.996(0.00)
RF	cpuSm	0.862(0.01)	0.855(0.01)	0.847(0.01)	0.854(0.01)	0.864(0.01)	0.858(0.01)	0.859(0.01)	0.865(0.01)	0.867(0.01)
	maxTorque	0.998(0.00)	0.973(0.01)	0.975(0.01)	0.976(0.01)	0.995(0.00)	0.994(0.00)	0.995(0.00)	0.995(0.00)	0.995(0.00)
	bank8FM	0.991(0.00)	0.979(0.00)	0.979(0.00)	0.977(0.00)	0.988(0.00)	0.987(0.00)	0.988(0.00)	0.988(0.00)	0.989(0.00)
	dAiler	0.946(0.00)	0.878(0.01)	0.871(0.01)	0.877(0.01)	0.906(0.01)	0.901(0.01)	0.905(0.01)	0.917(0.01)	0.921(0.00)
	concreteStrength	1.000(0.01)	0.966(0.02)	0.951(0.02)	0.954(0.02)	0.994(0.01)	0.991(0.01)	0.996(0.01)	0.994(0.01)	0.997(0.01)
	acceleration	0.996(0.01)	0.970(0.01)	0.970(0.01)	0.975(0.01)	0.989(0.01)	0.989(0.01)	0.989(0.01)	0.991(0.01)	0.991(0.01)
	airfoild	0.839(0.02)	0.811(0.02)	0.792(0.03)	0.808(0.02)	0.838(0.02)	0.845(0.01)	0.843(0.01)	0.842(0.02)	0.837(0.02)
	servo	0.909(0.05)	0.873(0.07)	0.894(0.06)	0.868(0.08)	0.878(0.06)	0.898(0.06)	0.879(0.08)	0.898(0.06)	0.883(0.07)
	a6	0.896(0.04)	0.819(0.06)	0.801(0.05)	0.817(0.05)	0.822(0.04)	0.817(0.04)	0.805(0.05)	0.828(0.05)	0.825(0.05)
	Abalone	0.913(0.01)	0.855(0.01)	0.863(0.01)	0.854(0.01)	0.838(0.01)	0.825(0.01)	0.830(0.01)	0.848(0.01)	0.851(0.01)
	machineCpu	0.974(0.01)	0.949(0.02)	0.963(0.02)	0.949(0.02)	0.959(0.02)	0.955(0.02)	0.953(0.03)	0.962(0.02)	0.959(0.02)
	a3	0.918(0.03)	0.787(0.07)	0.797(0.06)	0.769(0.07)	0.820(0.04)	0.816(0.07)	0.815(0.06)	0.812(0.06)	0.816(0.06)
	a4	0.895(0.05)	0.811(0.07)	0.811(0.06)	0.813(0.06)	0.831(0.06)	0.827(0.05)	0.818(0.06)	0.837(0.05)	0.831(0.05)
	a1	0.941(0.03)	0.830(0.07)	0.843(0.06)	0.824(0.07)	0.854(0.07)	0.858(0.06)	0.850(0.07)	0.874(0.06)	0.869(0.06)
	a7	0.915(0.03)	0.780(0.06)	0.789(0.07)	0.784(0.07)	0.816(0.07)	0.815(0.07)	0.811(0.07)	0.807(0.07)	0.804(0.07)
SVM	boston	0.985(0.01)	0.950(0.02)	0.952(0.02)	0.948(0.02)	0.959(0.01)	0.955(0.01)	0.957(0.02)	0.962(0.01)	0.963(0.01)
	a2	0.952(0.02)	0.801(0.07)	0.816(0.06)	0.821(0.07)	0.846(0.05)	0.842(0.06)	0.841(0.06)	0.856(0.05)	0.851(0.05)
	a5	0.947(0.03)	0.817(0.07)	0.826(0.06)	0.830(0.06)	0.844(0.05)	0.850(0.05)	0.842(0.05)	0.853(0.05)	0.837(0.06)
	fuelCons	0.970(0.01)	0.947(0.01)	0.947(0.01)	0.944(0.01)	0.954(0.01)	0.951(0.01)	0.951(0.01)	0.957(0.01)	0.957(0.01)
	availPwr	0.992(0.00)	0.979(0.01)	0.981(0.00)	0.980(0.01)	0.984(0.00)	0.979(0.00)	0.978(0.01)	0.985(0.00)	0.985(0.00)
	cpuSm	0.863(0.01)	0.800(0.01)	0.800(0.01)	0.800(0.01)	0.819(0.01)	0.840(0.01)	0.840(0.01)	0.820(0.01)	0.821(0.01)
	maxTorque	0.996(0.00)	0.984(0.00)	0.985(0.00)	0.983(0.00)	0.989(0.00)	0.989(0.00)	0.989(0.00)	0.989(0.00)	0.988(0.00)
	bank8FM	0.997(0.00)	0.990(0.00)	0.989(0.00)	0.990(0.00)	0.989(0.00)	0.988(0.00)	0.988(0.00)	0.990(0.00)	0.991(0.00)
	dAiler	0.956(0.00)	0.879(0.01)	0.883(0.01)	0.879(0.01)	0.851(0.01)	0.845(0.01)	0.846(0.01)	0.875(0.01)	0.876(0.01)

Continued on next page

Table A.12 – continued from previous page

Learner	Data sets	None	U._._	U.F._	U.S._	S._._	S.F.F	S.S.F	S.F.S	S.S.S
SVM	concreteStrength	0.992(0.01)	0.944(0.02)	0.941(0.02)	0.943(0.02)	0.955(0.01)	0.954(0.01)	0.954(0.01)	0.961(0.01)	0.960(0.01)
	acceleration	0.995(0.01)	0.957(0.01)	0.962(0.01)	0.956(0.01)	0.961(0.01)	0.953(0.01)	0.952(0.01)	0.971(0.01)	0.970(0.01)
	airfoild	0.853(0.02)	0.792(0.03)	0.776(0.03)	0.794(0.03)	0.823(0.02)	0.823(0.02)	0.825(0.02)	0.826(0.02)	0.827(0.02)
NNET	servo	0.934(0.03)	0.802(0.21)	0.877(0.19)	0.807(0.20)	0.848(0.25)	0.872(0.21)	0.910(0.17)	0.849(0.21)	0.851(0.21)
	a6	0.814(0.05)	0.488(0.26)	0.518(0.29)	0.539(0.28)	0.617(0.28)	0.629(0.30)	0.645(0.30)	0.641(0.30)	0.602(0.30)
	Abalone	0.910(0.01)	0.851(0.02)	0.848(0.02)	0.854(0.02)	0.847(0.02)	0.830(0.02)	0.837(0.02)	0.862(0.02)	0.865(0.02)
	machineCpu	0.844(0.04)	0.367(0.43)	0.326(0.38)	0.395(0.44)	0.484(0.42)	0.655(0.29)	0.655(0.23)	0.424(0.29)	0.416(0.40)
	a3	0.840(0.06)	0.630(0.19)	0.646(0.20)	0.649(0.21)	0.689(0.25)	0.670(0.26)	0.671(0.27)	0.644(0.29)	0.687(0.27)
	a4	0.869(0.06)	0.620(0.21)	0.570(0.21)	0.640(0.21)	0.706(0.20)	0.676(0.21)	0.655(0.23)	0.724(0.18)	0.690(0.21)
	a1	0.937(0.04)	0.792(0.11)	0.809(0.09)	0.829(0.10)	0.819(0.11)	0.786(0.11)	0.803(0.11)	0.816(0.12)	0.816(0.12)
	a7	0.843(0.10)	0.631(0.28)	0.622(0.28)	0.656(0.26)	0.683(0.28)	0.687(0.28)	0.674(0.29)	0.717(0.27)	0.739(0.24)
	boston	1.000(0.03)	0.777(0.10)	0.782(0.13)	0.803(0.11)	0.777(0.10)	0.795(0.11)	0.804(0.12)	0.781(0.13)	0.791(0.11)
	a2	0.913(0.04)	0.729(0.11)	0.752(0.12)	0.778(0.12)	0.785(0.13)	0.777(0.14)	0.778(0.14)	0.792(0.12)	0.775(0.12)
	a5	0.901(0.03)	0.713(0.21)	0.767(0.16)	0.786(0.14)	0.803(0.15)	0.806(0.15)	0.788(0.15)	0.790(0.17)	0.778(0.18)
	fuelCons	0.946(0.01)	0.850(0.05)	0.826(0.06)	0.841(0.05)	0.870(0.07)	0.860(0.08)	0.862(0.08)	0.865(0.06)	0.854(0.07)
	availPwr	1.000(0.02)	0.663(0.11)	0.651(0.09)	0.660(0.09)	0.606(0.07)	0.631(0.07)	0.631(0.07)	0.590(0.07)	0.589(0.06)
	cpuSm	0.823(0.06)	0.487(0.21)	0.469(0.21)	0.498(0.21)	0.485(0.23)	0.551(0.11)	0.561(0.10)	0.546(0.21)	0.536(0.19)
	maxTorque	1.000(0.01)	0.761(0.07)	0.745(0.08)	0.756(0.08)	0.722(0.07)	0.738(0.10)	0.715(0.06)	0.719(0.08)	0.721(0.08)
	bank8FM	0.991(0.00)	0.969(0.03)	0.969(0.03)	0.970(0.03)	0.985(0.01)	0.977(0.02)	0.980(0.02)	0.978(0.03)	0.983(0.02)
	dAiler	0.952(0.01)	0.884(0.03)	0.883(0.02)	0.885(0.02)	0.883(0.02)	0.882(0.02)	0.883(0.02)	0.882(0.02)	0.885(0.02)
	concreteStrength	1.000(0.01)	0.854(0.06)	0.815(0.06)	0.856(0.06)	0.808(0.05)	0.803(0.06)	0.813(0.05)	0.819(0.08)	0.829(0.08)
	acceleration	1.000(0.02)	0.760(0.06)	0.748(0.06)	0.775(0.06)	0.775(0.07)	0.796(0.06)	0.811(0.06)	0.780(0.09)	0.775(0.08)
	airfoild	0.894(0.02)	0.608(0.09)	0.588(0.10)	0.611(0.09)	0.581(0.07)	0.597(0.08)	0.594(0.06)	0.599(0.11)	0.594(0.09)

A.3 Evaluation Results of Pre-processing Strategies with Different Data Distribution Modifications Applied

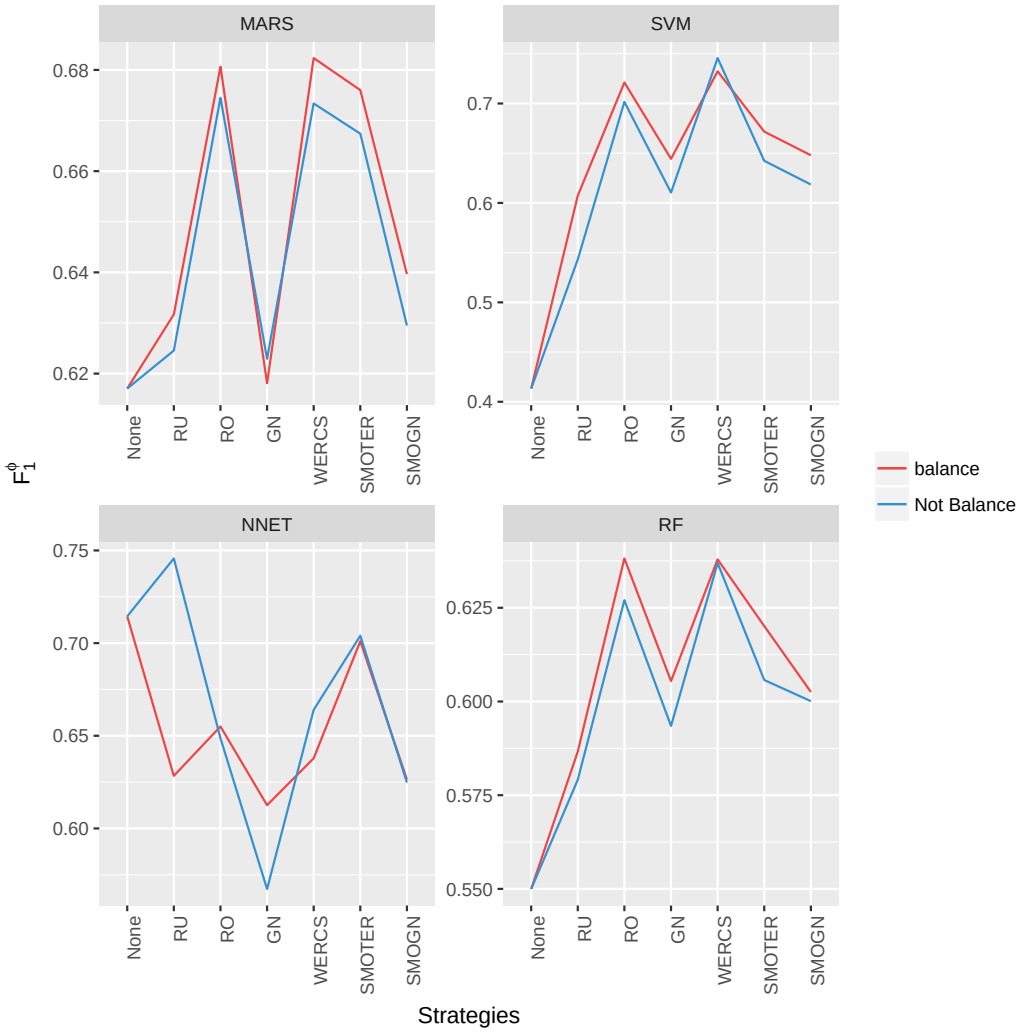


Figure A.1: Results of F_1^ϕ measure on *servo* data set, by learner, with pre-processing strategies set to either balancing or to consider other not balancing variants.

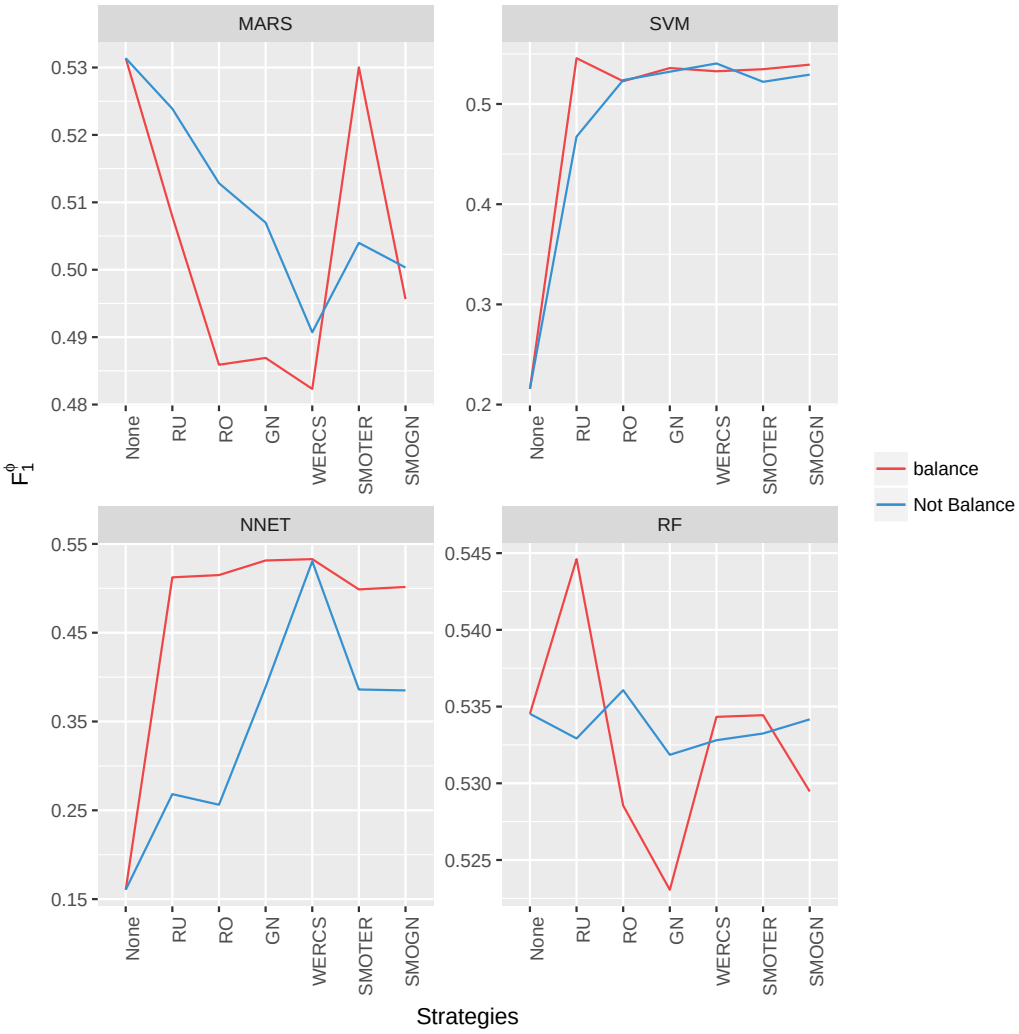


Figure A.2: Results of F_1^ϕ measure on *a6* data set, by learner, with pre-processing strategies set to either balancing or to consider other not balancing variants.

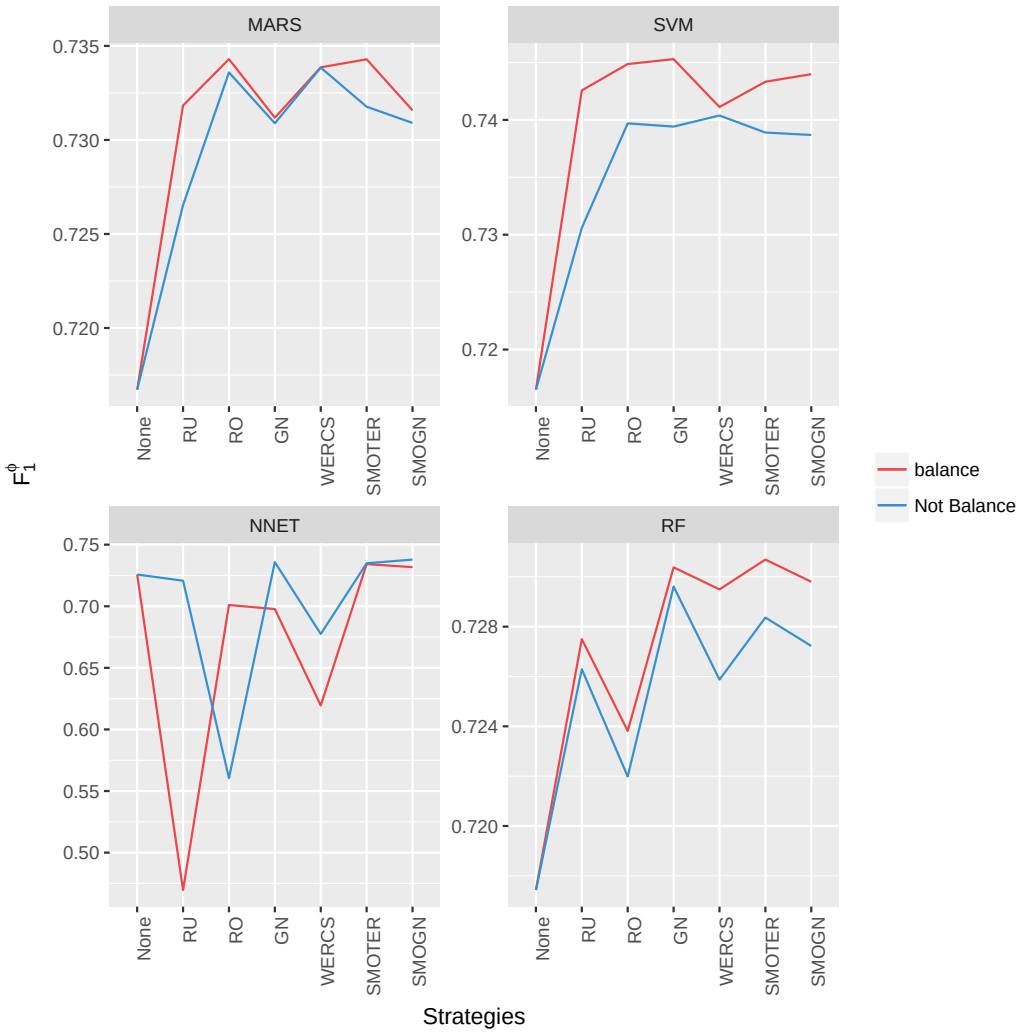


Figure A.3: Results of F_1^ϕ measure on *Abalone* data set, by learner, with pre-processing strategies set to either balancing or to consider other not balancing variants.

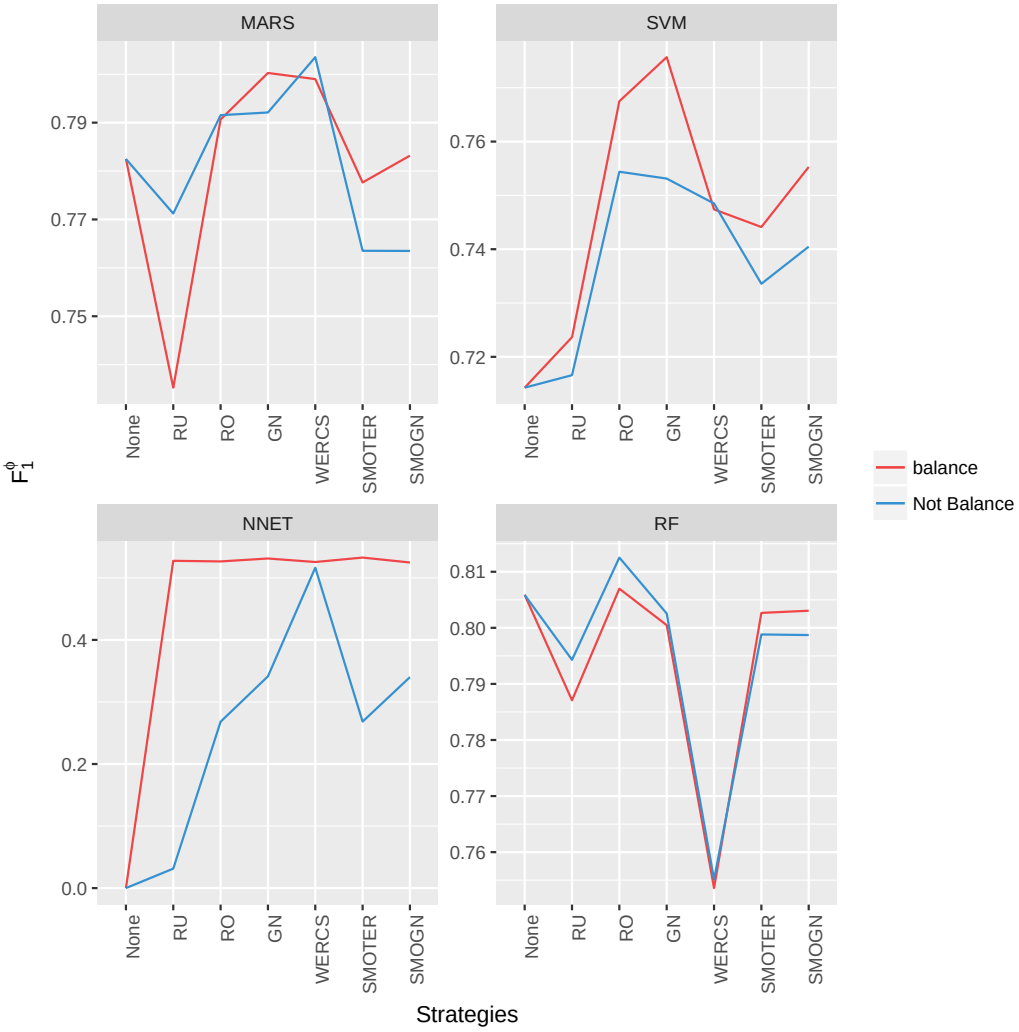


Figure A.4: Results of F_1^ϕ measure on *machineCpu* data set, by learner, with pre-processing strategies set to either balancing or to consider other not balancing variants.

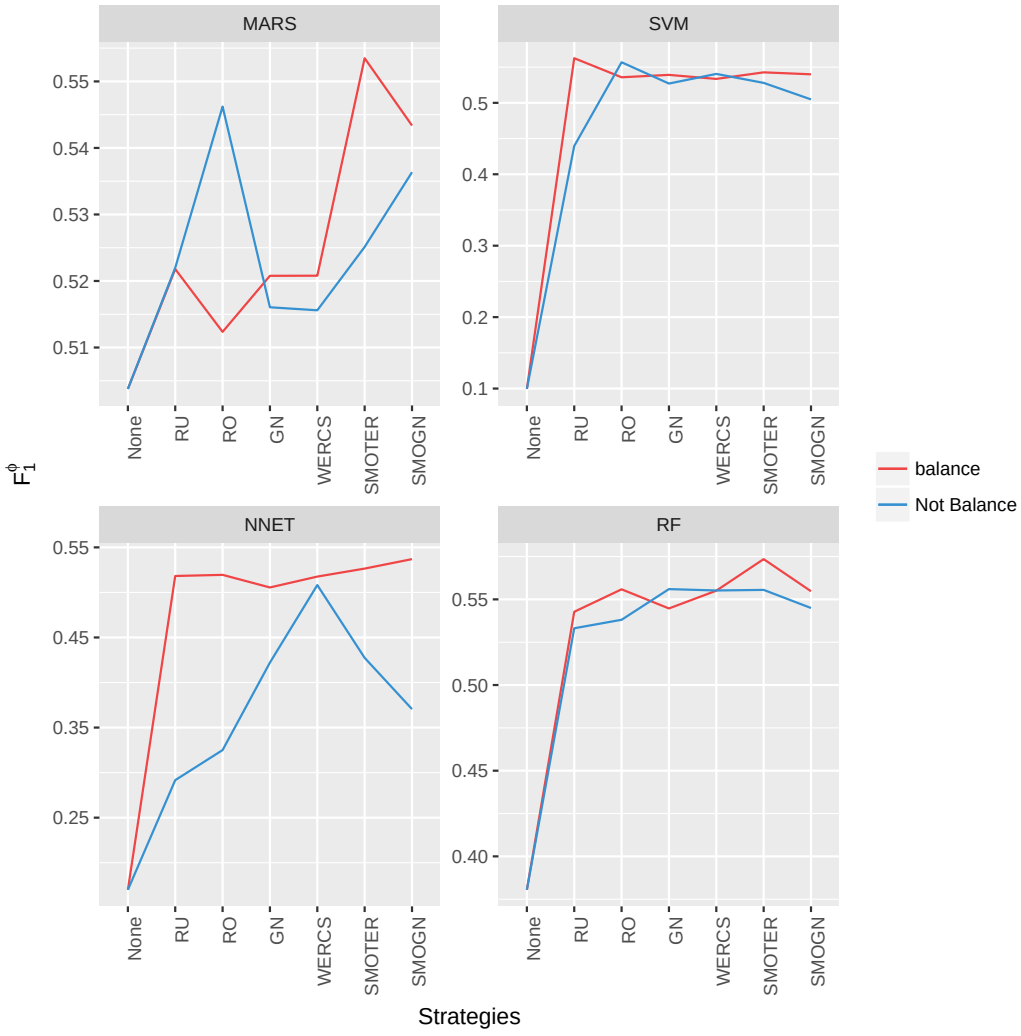


Figure A.5: Results of F_1^ϕ measure on *a3* data set, by learner, with pre-processing strategies set to either balancing or to consider other not balancing variants.

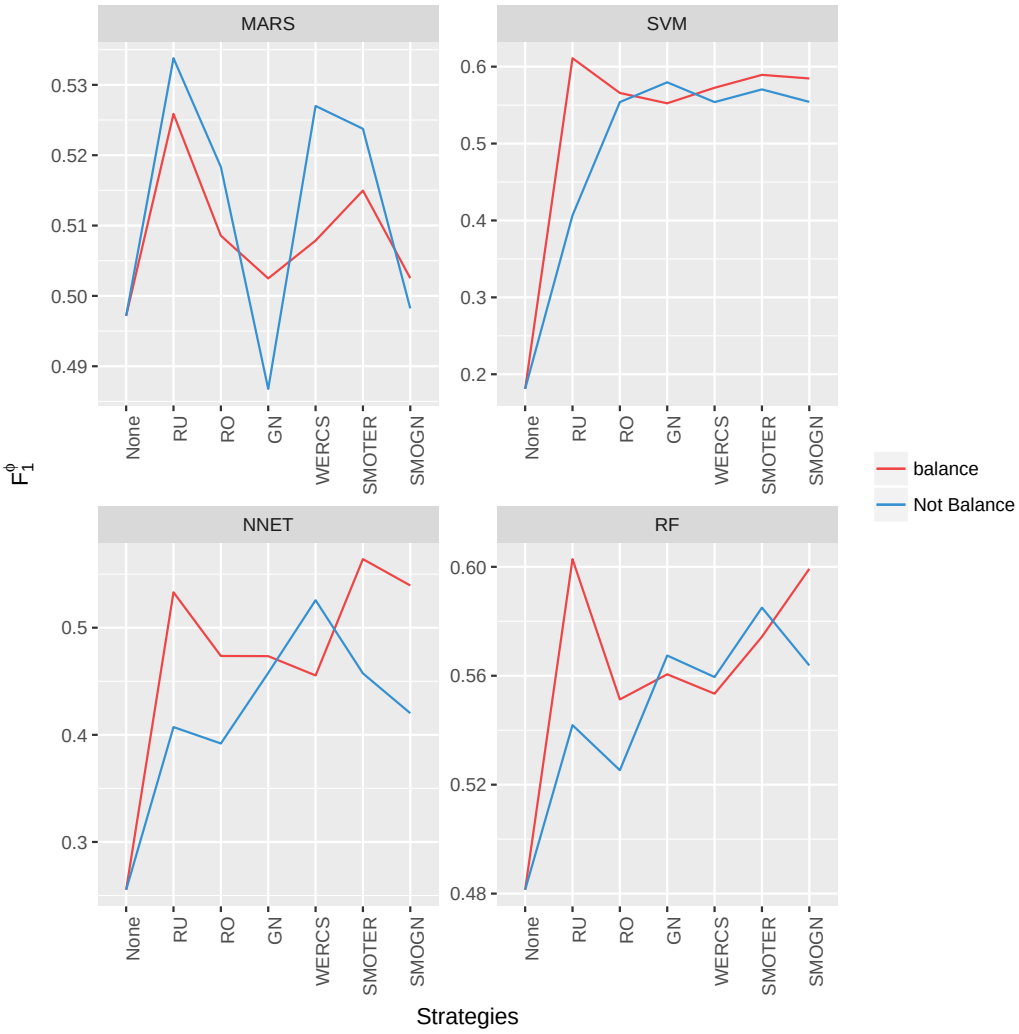


Figure A.6: Results of F_1^ϕ measure on *a4* data set, by learner, with pre-processing strategies set to either balancing or to consider other not balancing variants.

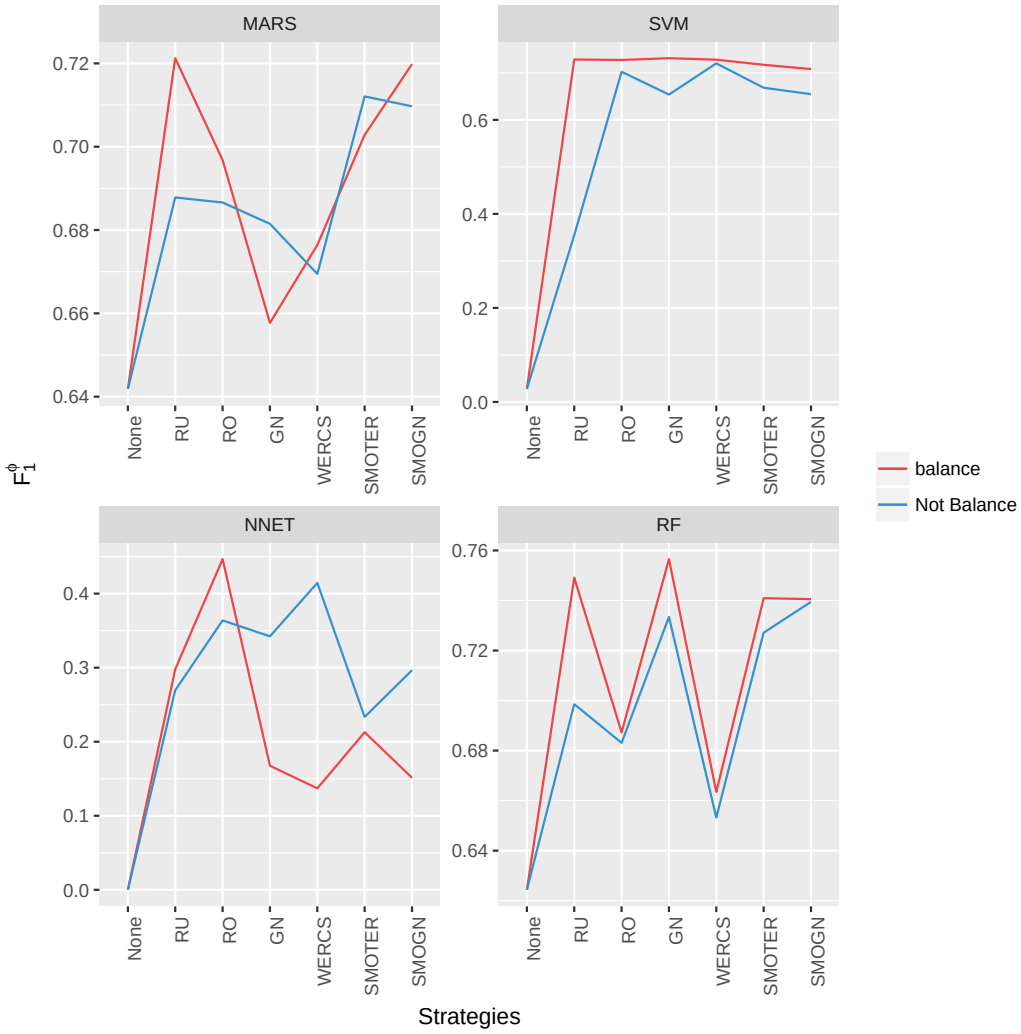


Figure A.7: Results of F_1^ϕ measure on *a1* data set, by learner, with pre-processing strategies set to either balancing or to consider other not balancing variants.

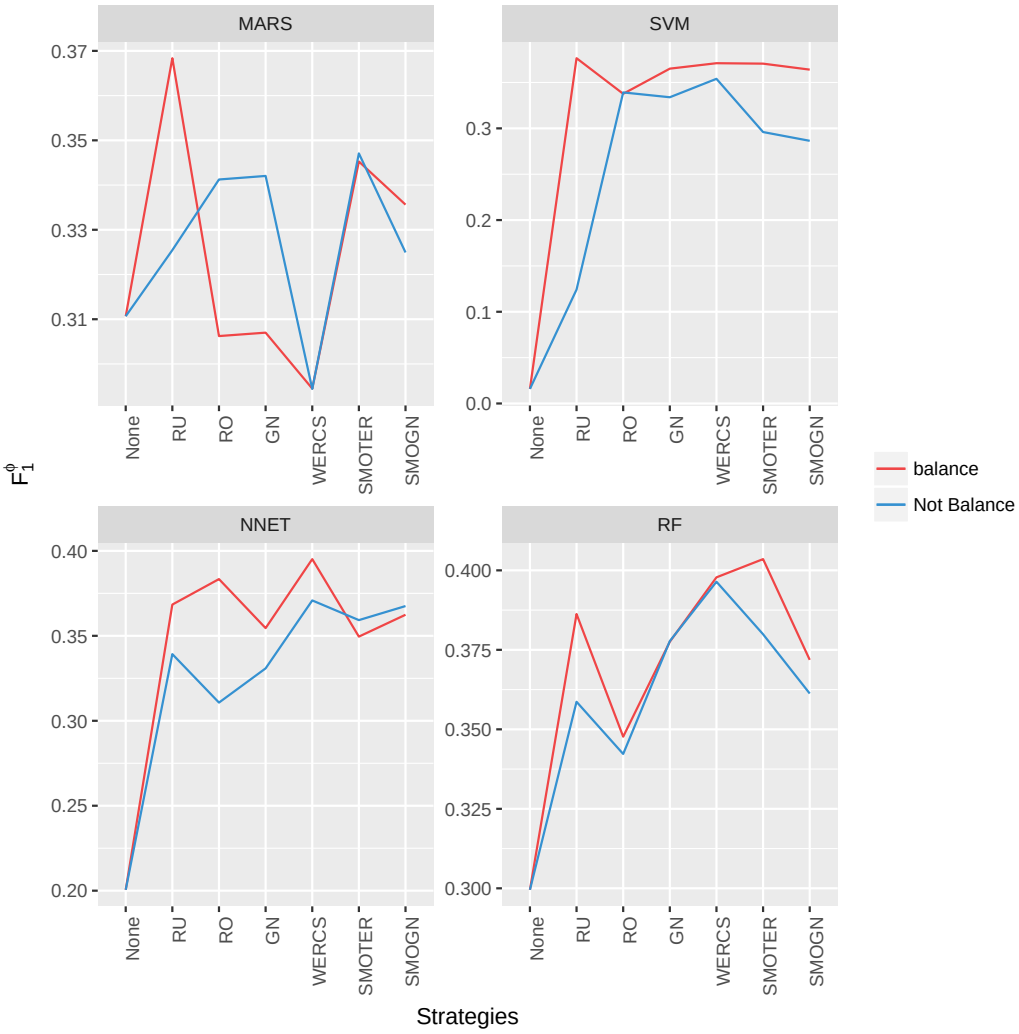


Figure A.8: Results of F_1^ϕ measure on *a7* data set, by learner, with pre-processing strategies set to either balancing or to consider other not balancing variants.

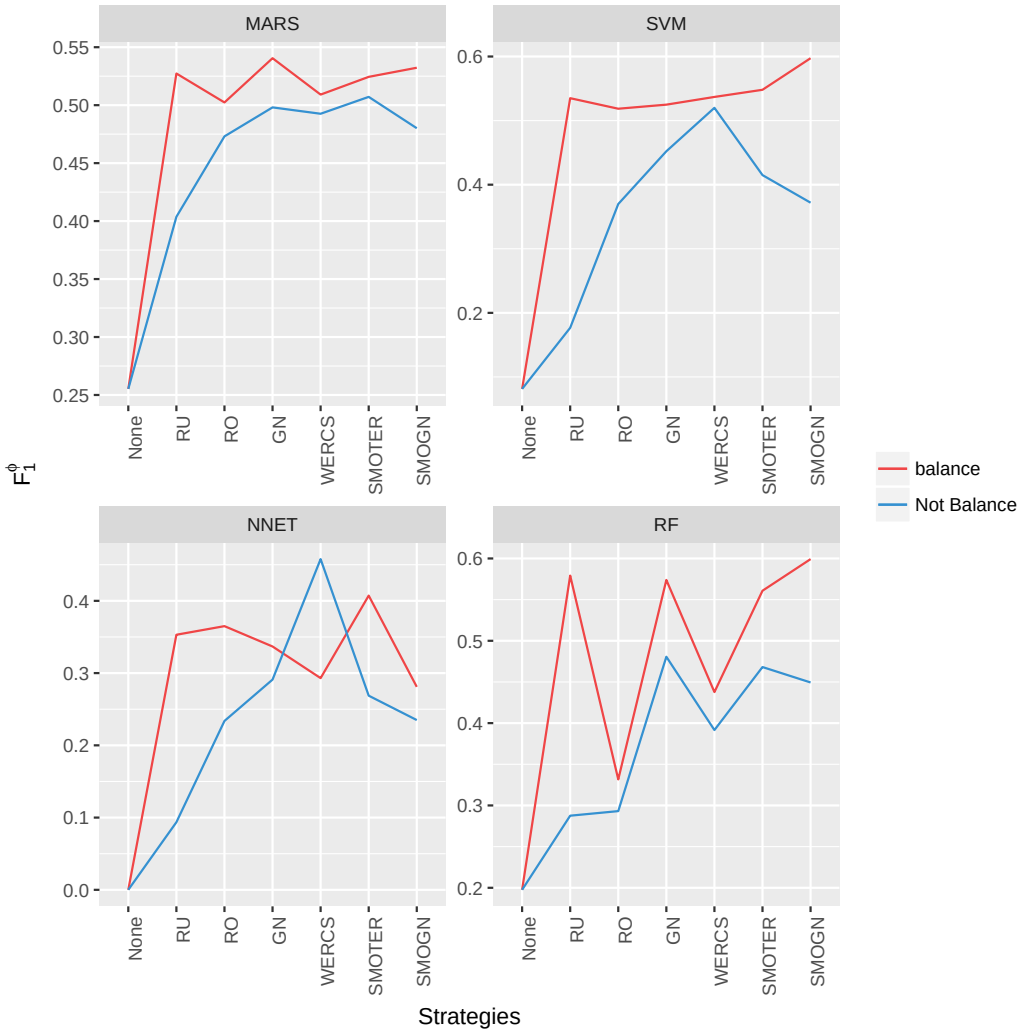


Figure A.9: Results of F_1^ϕ measure on *a2* data set, by learner, with pre-processing strategies set to either balancing or to consider other not balancing variants.

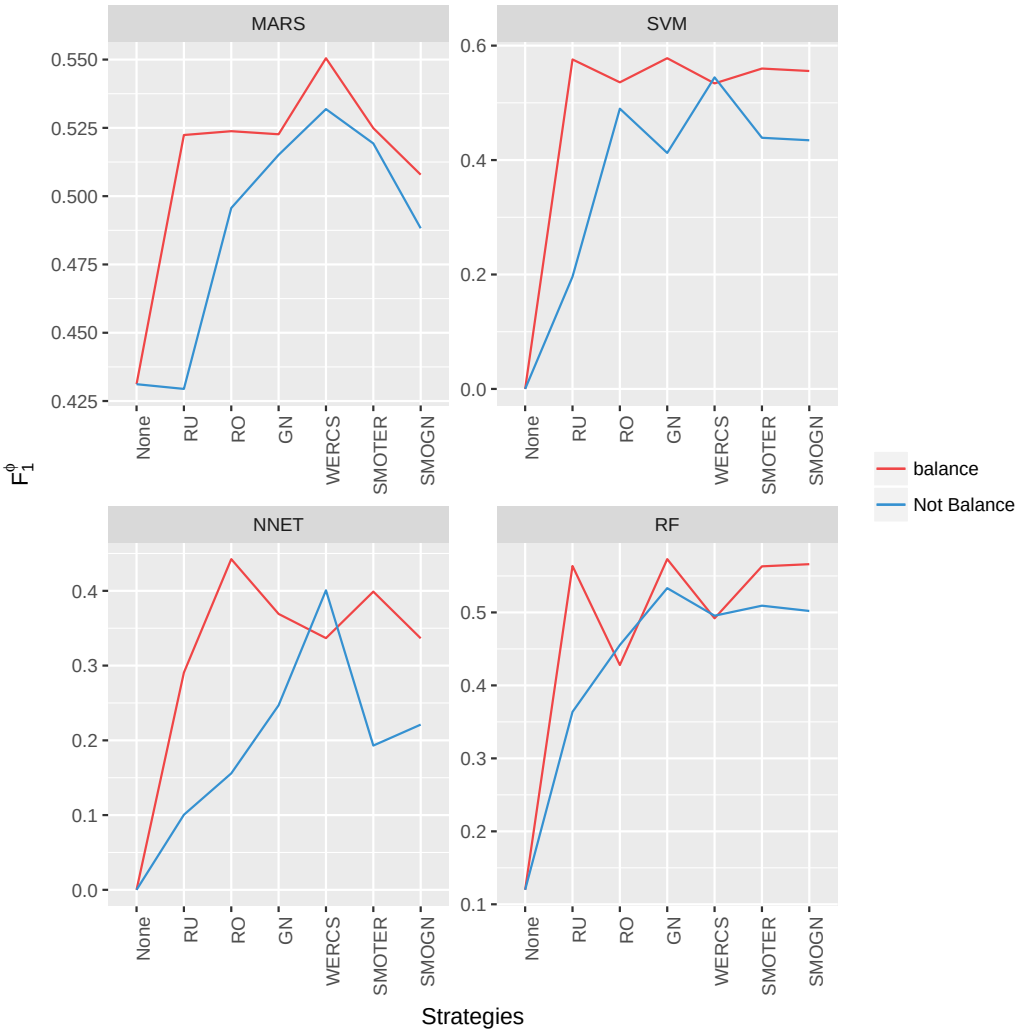


Figure A.10: Results of F_1^ϕ measure on *a5* data set, by learner, with pre-processing strategies set to either balancing or to consider other not balancing variants.

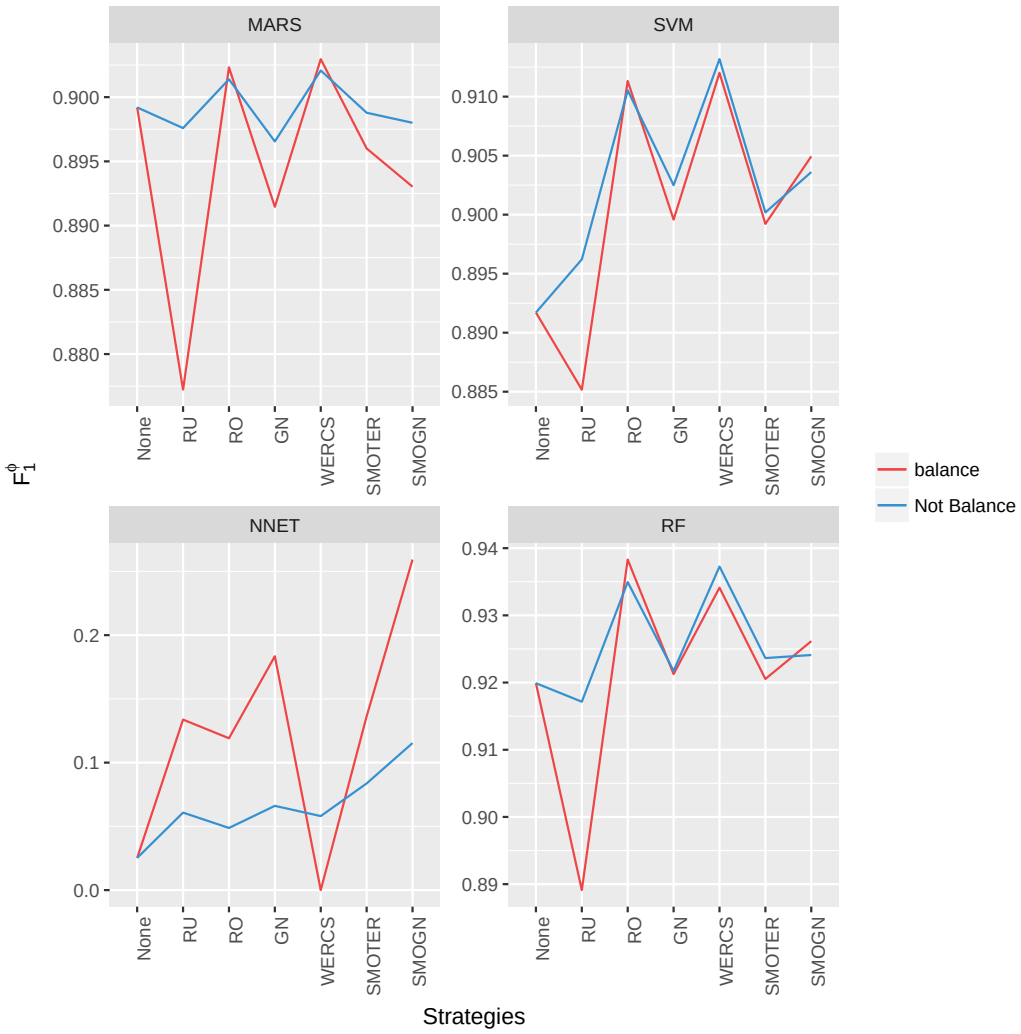


Figure A.11: Results of F_1^ϕ measure on *fuelCons* data set, by learner, with pre-processing strategies set to either balancing or to consider other not balancing variants.

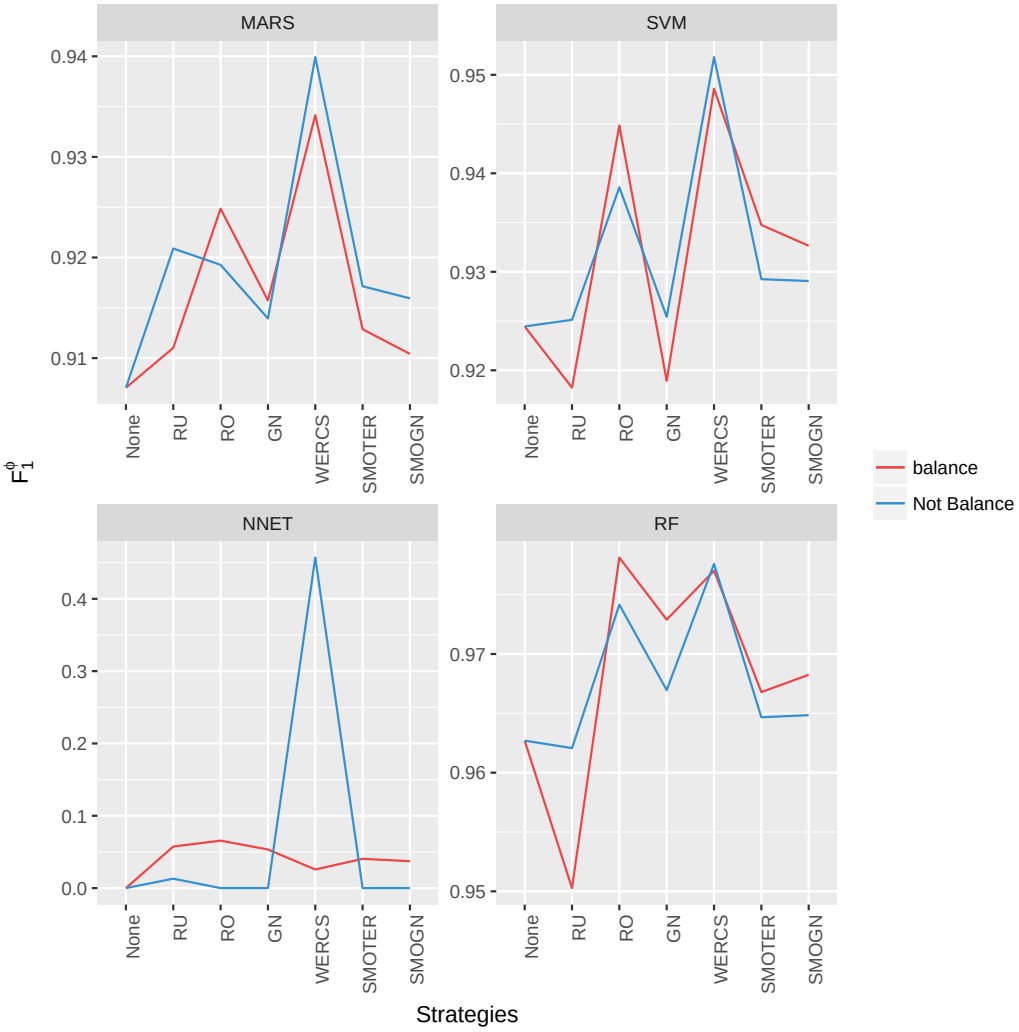


Figure A.12: Results of F_1^ϕ measure on *availPwr* data set, by learner, with pre-processing strategies set to either balancing or to consider other not balancing variants.

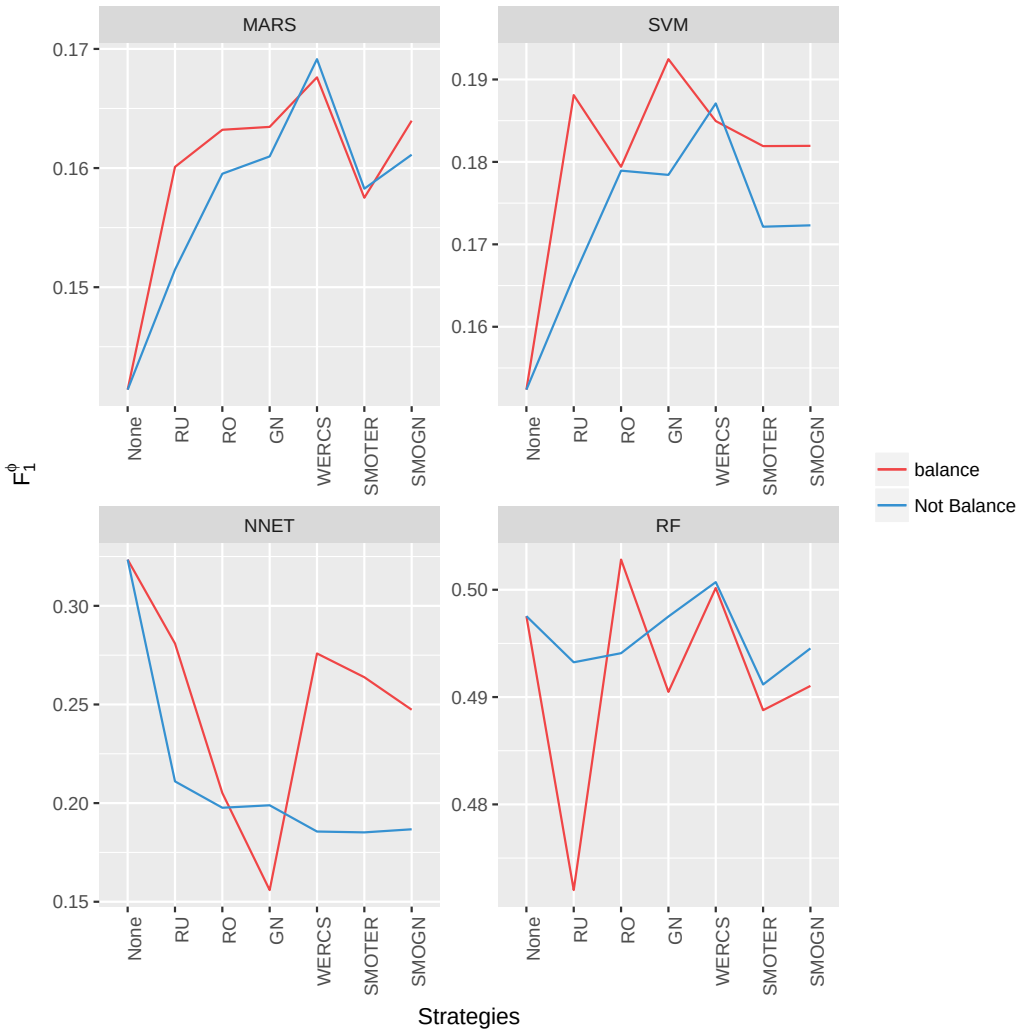


Figure A.13: Results of F_1^ϕ measure on *cpuSm* data set, by learner, with pre-processing strategies set to either balancing or to consider other not balancing variants.

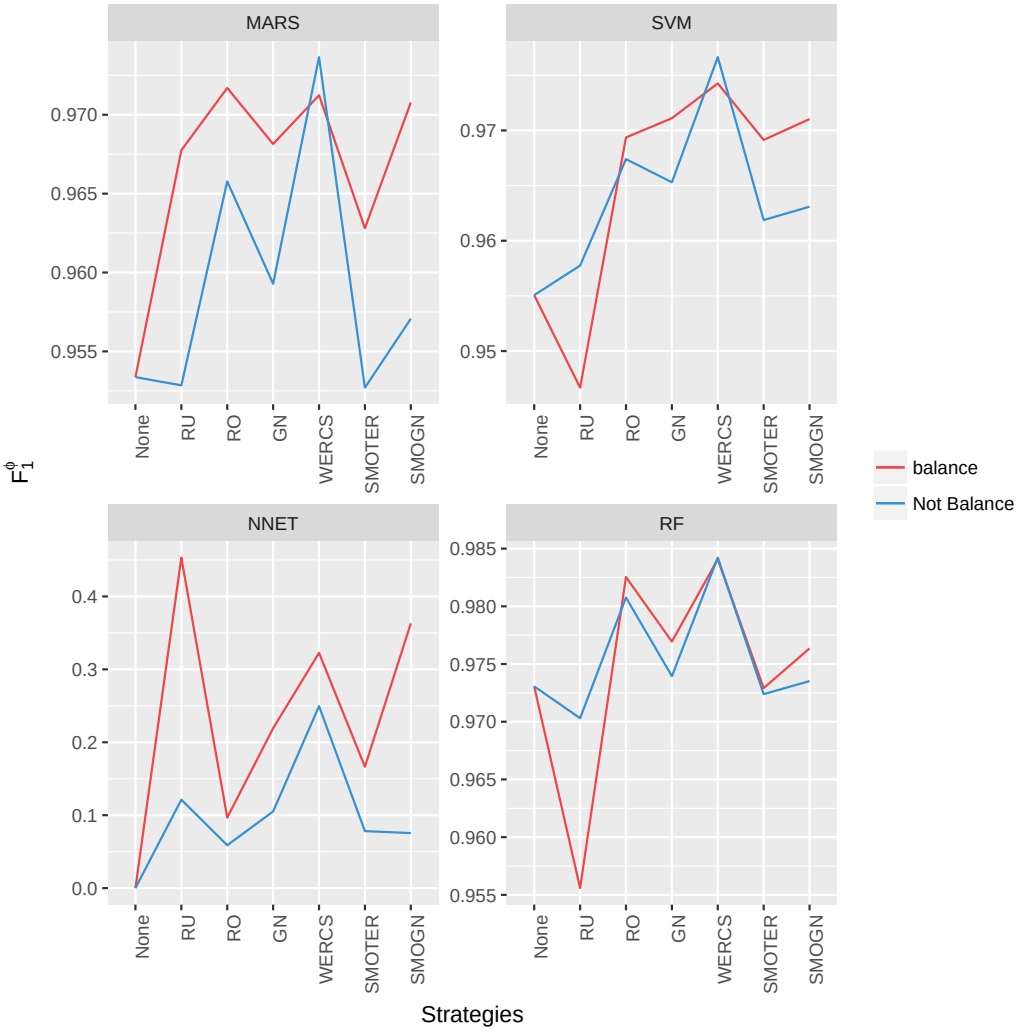


Figure A.14: Results of F_1^ϕ measure on *maxTorque* data set, by learner, with pre-processing strategies set to either balancing or to consider other not balancing variants.

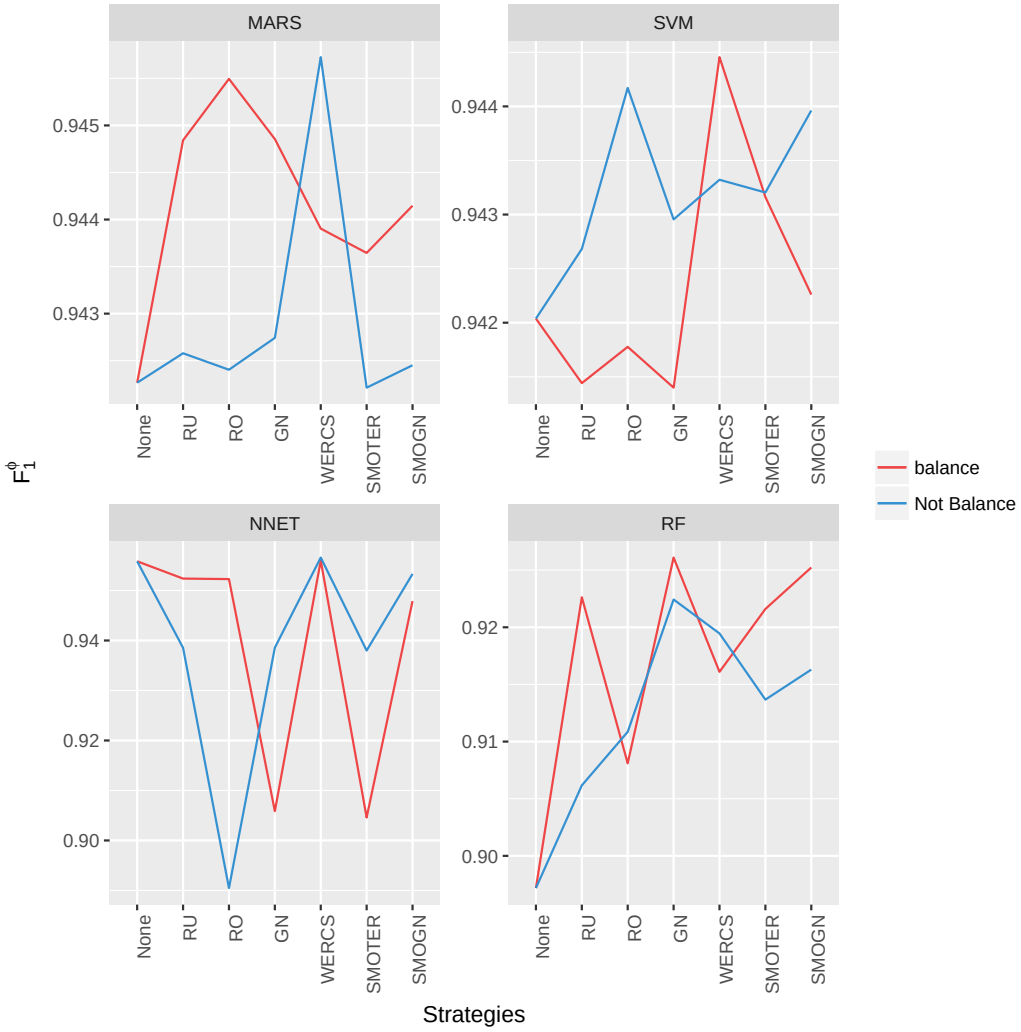


Figure A.15: Results of F_1^ϕ measure on *bank8FM* data set, by learner, with pre-processing strategies set to either balancing or to consider other not balancing variants.

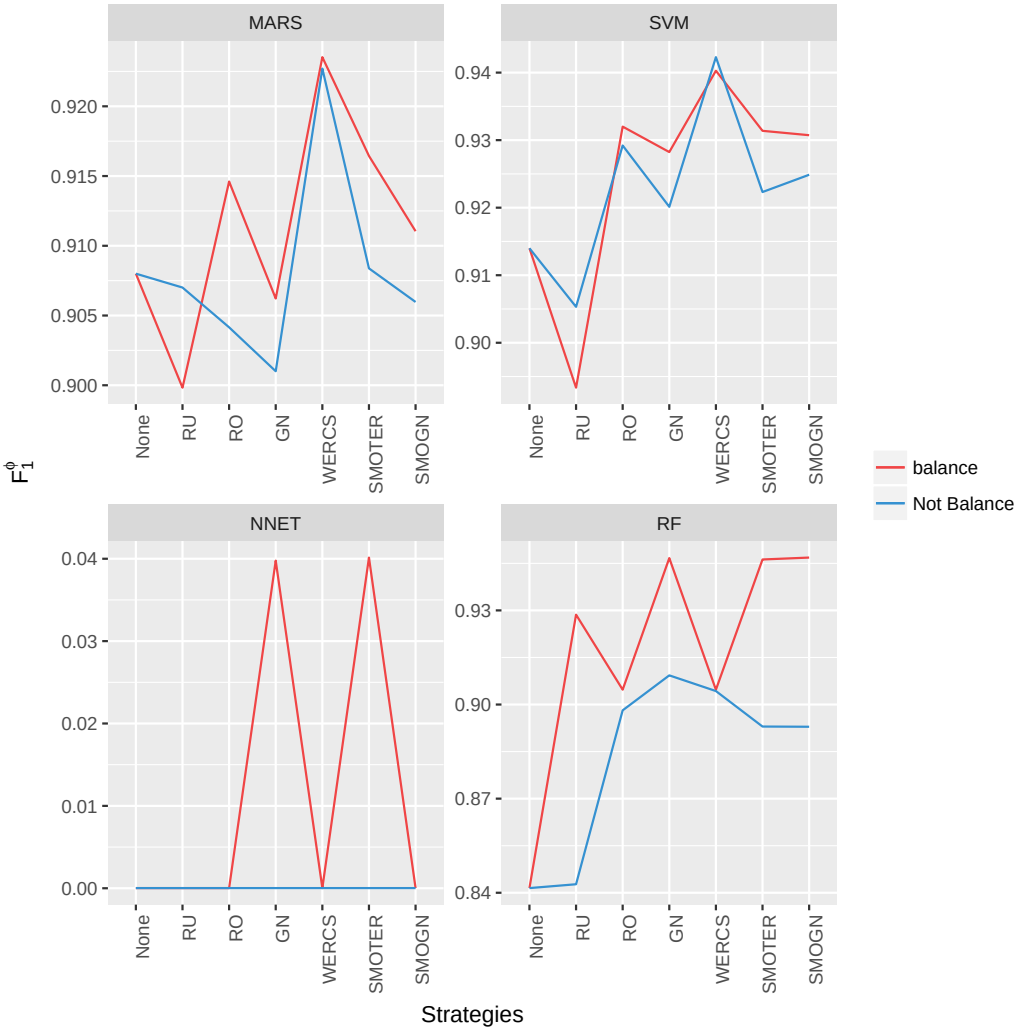


Figure A.16: Results of F_1^ϕ measure on *concreteStrength* data set, by learner, with pre-processing strategies set to either balancing or to consider other not balancing variants.

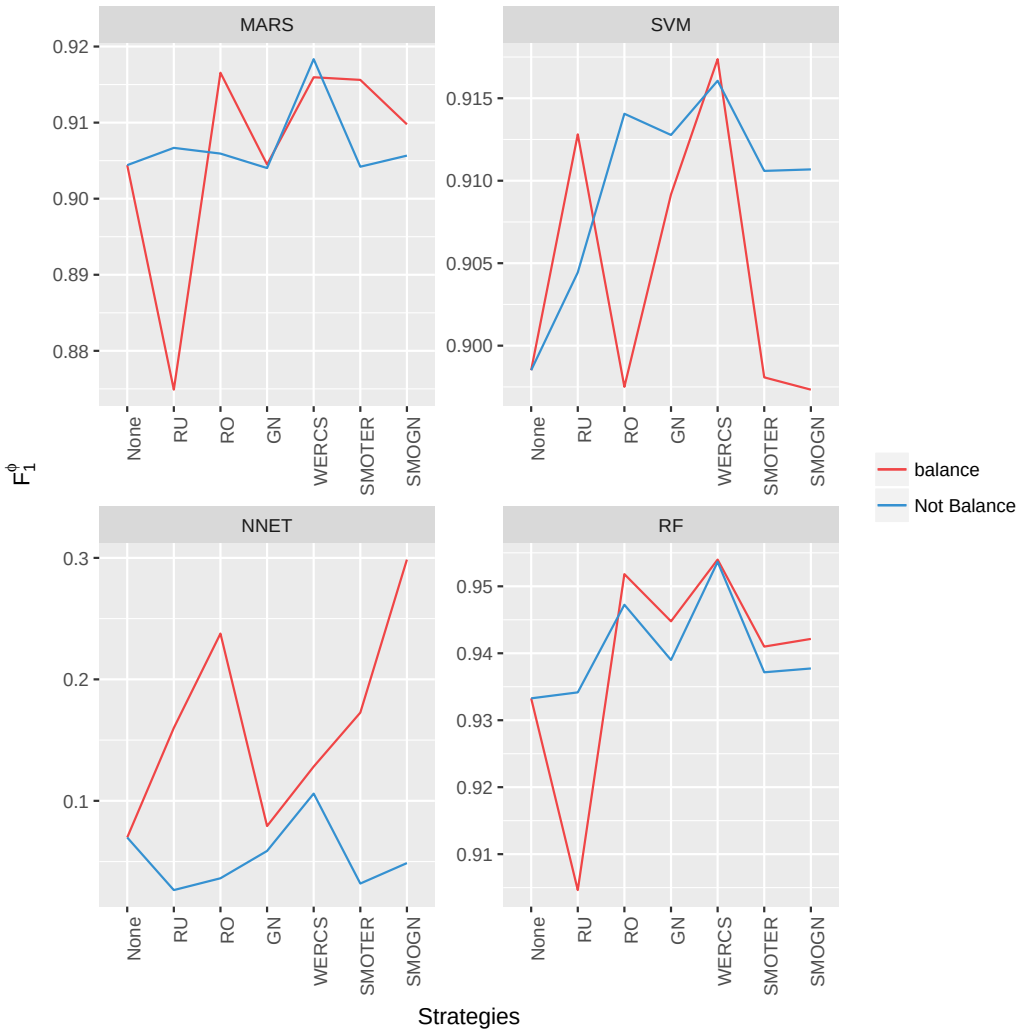


Figure A.17: Results of F_1^ϕ measure on *acceleration* data set, by learner, with pre-processing strategies set to either balancing or to consider other not balancing variants.

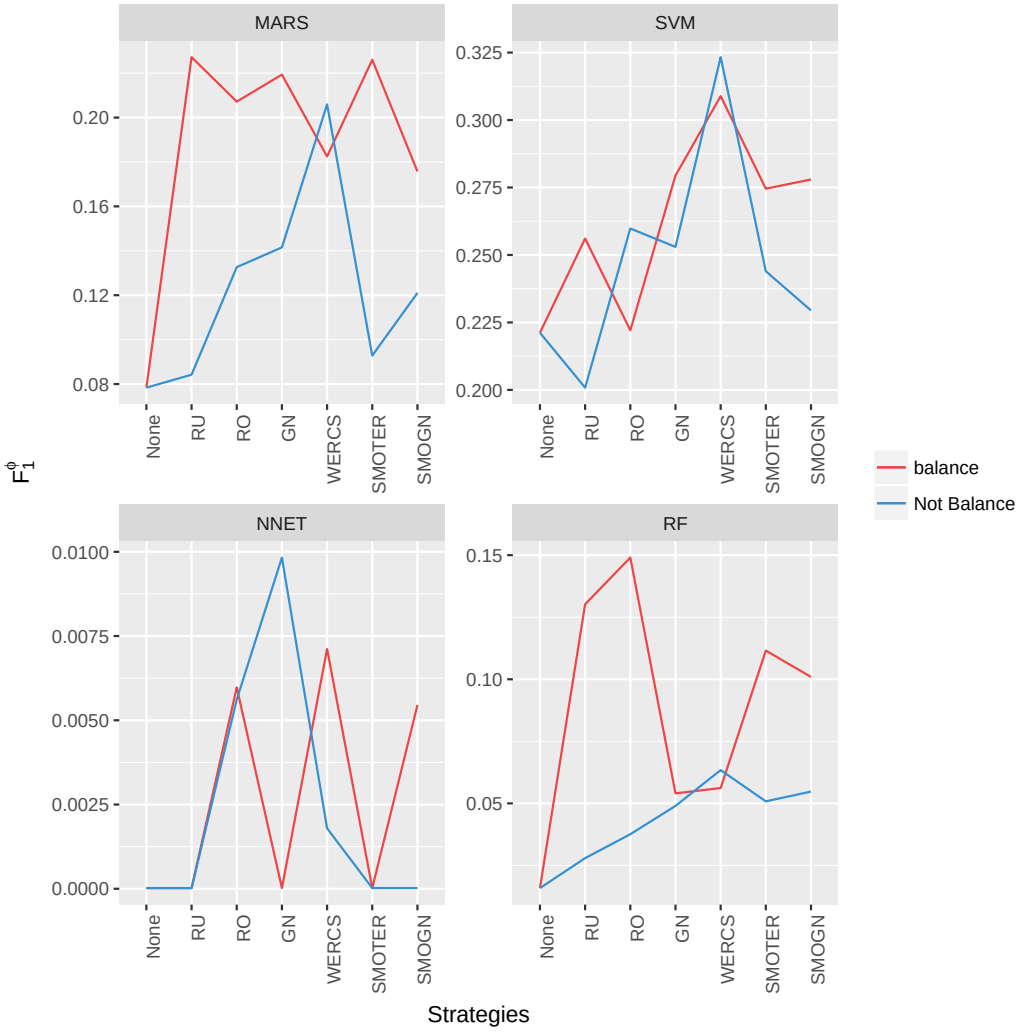


Figure A.18: Results of F_1^ϕ measure on *airfoild* data set, by learner, with pre-processing strategies set to either balancing or to consider other not balancing variants.

Glossary

AOC Area Over the RROC Curve. 24

AUC-PR Area Under the PR Curve. xxi, 42, 44

AUC-ROC Area Under the ROC Curve. xxi, 40–42, 44, 94

AUC-ROCIV Area Under the ROCIV Curve. 41, 44

CBO Cluster-Based Over-sampling. 52

CWA *Mean Class Weighted Accuracy*. 39

EA Evolutionary Algorithm. 52

G-Mean *Geometric-Mean*. 38, 39, 94, 95

GA Genetic Algorithm. 31

GN Introduction of Gaussian Noise. xxii, 127, 128, 132, 133, 138, 143, 144, 155

k-NN k-Nearest Neighbours. 49

MAE Mean Absolute Error. 43, 97

MC Mean Cost. 22, 23

MSE Mean Squared Error. 43, 97

MU Mean Utility. 23, 24, 43, 88

NMC Normalised Mean Cost. 23

NMU Normalised Mean Utility. 43, 74, 88, 109

NNET Neural Network. 28, 30, 31, 48, 54, 153

OSS One Sided Selection. 52

PR Curve Precision-Recall Curve. xxi, 41, 42

REC Curve Regression Error Characteristic Curve. xxi, 24, 26, 44, 45

REC Surface Regression Error Characteristic Surface. xxi, 45

RO Random Over-sampling. xxii, 122, 123, 125, 143, 144, 155

ROC Receiver Operating Characteristics curve. xxi, 24, 40–42

ROCIV Instance Varying Receiver Operating Characteristics curve. 41

ROSE Random Over Sampling Examples. 53

RROC Curve ROC Space for Regression. xxi, 24, 25

RU Random Under-sampling. xxii, 121–123, 143, 144, 149, 155

SMOGN SMOTER with Gaussian Noise. xxii, 132, 133, 135, 143, 144, 155

SMOTE Synthetic Minority Over-sampling TEchnique. xxi, 53, 54, 128, 130

SMOTER SMOTE for Regression. xxii, 128, 130, 132, 133, 138, 143, 144, 149, 155

SVM Support Vector Machine. 28, 31, 48, 49, 52, 54

UBDM Utility Based Data Mining. 9, 10

WERCS **WE**ighted **R**elevance-based **C**ombination **S**trategy. xxii, 123, 125–127, 143, 144, 153, 155

References

Naoki Abe, Bianca Zadrozny, and John Langford. An iterative method for multi-class cost-sensitive learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 3–11. ACM, 2004.

Ni Ailing, Shujie Yang, Xiaofeng Zhu, and Shichao Zhang. Learning classification rules under multiple costs. *Asian Journal of Information Technology*, 4(11):1080–1085, 2005.

Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In *Machine Learning: ECML 2004*, pages 39–50. Springer, 2004.

Roberto Alejo, Vicente García, José Martínez Sotoca, Ramón Alberto Mollineda, and José Salvador Sánchez. Improving the performance of the rbf neural networks trained with imbalanced samples. In *Computational and Ambient Intelligence*, pages 162–169. Springer, 2007.

Roberto Alejo, J. A Antonio, Rosa Maria Valdovinos, and J. Horacio Pacheco-Sánchez. Assessments metrics for multi-class imbalance learning: A preliminary study. In *Pattern Recognition*, pages 335–343. Springer, 2013.

Shun-ichi Amari and Si Wu. Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, 12(6):783–789, 1999.

Francis R Bach, David Heckerman, and Eric Horvitz. Considering cost asymmetry in learning classifiers. *Journal of Machine Learning Research*, 7(Aug):1713–1741, 2006.

Alejandro Correa Bahnsen, Djamila Aouada, and Björn Ottersten. Example-dependent cost-sensitive decision trees. *Expert Systems with Applications*, 42(19):6609–6619, 2015.

Luís Carlos Gouveia Baía. Actionable forecasting and activity monitoring: applications to financial trading. Master's thesis, Faculty of Sciences - University of Porto, 2015.

Gaurav Bansal, Atish P. Sinha, and Huimin Zhao. Tuning data mining methods for cost-sensitive regression: a study in loan charge-off forecasting. *Journal of Management Information Systems*, 25(3):315–336, 2008.

Ricardo Barandela, José Salvador Sánchez, Vicente Garcıa, and Edgar Rangel. Strategies for learning in class imbalance problems. *Pattern Recognition*, 36(3):849–851, 2003.

Vincent Barnab-Lortie, Colin Bellinger, and Nathalie Japkowicz. Active learning for one-class classification. In *Proceedings of ICMLA'2015*, 2015.

Sukarna Barua, Monirul Islam, Xin Yao, and Kazuyuki Murase. Mwmote-majority weighted minority oversampling technique for imbalanced data set learning. 2012.

Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1):20–29, 2004.

Rukshan Batuwita and Vasile Palade. A new performance measure for class imbalance learning. application to bioinformatics problems. In *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*, pages 545–550. IEEE, 2009.

Rukshan Batuwita and Vasile Palade. Efficient resampling methods for training support vector machines with imbalanced datasets. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE, 2010a.

Rukshan Batuwita and Vasile Palade. Fsvm-cil: fuzzy support vector machines for class imbalance learning. *Fuzzy Systems, IEEE Transactions on*, 18(3):558–571, 2010b.

Rukshan Batuwita and Vasile Palade. Adjusted geometric-mean: a novel performance measure for imbalanced bioinformatics datasets learning. *Journal of Bioinformatics and Computational Biology*, 10(04), 2012.

Colin Bellinger, Shiven Sharma, and Nathalie Japkowicz. One-class versus binary classification: Which and when? In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 102–106. IEEE, 2012.

Colin Bellinger, Nathalie Japkowicz, and Christopher Drummond. Synthetic oversampling for advanced radioactive threat detection. In *Proceedings ICML'2015*, 2015.

Colin Bellinger, Christopher Drummond, and Nathalie Japkowicz. Manifold-based synthetic oversampling with manifold conformance estimation. *Machine Learning*, 107(3):605–637, 2018.

Jinbo Bi and Kristin P Bennett. Regression error characteristic curves. In *Proc. of the 20th Int. Conf. on Machine Learning*, pages 43–50, 2003.

Jerzy Błaszczyński and Jerzy Stefanowski. Neighbourhood sampling in bagging for imbalanced data. *Neurocomputing*, 150:529–542, 2015.

Adam Blum. Neural networks in c++. *NY: Wiley*, 697, 1992.

Jeffrey P Bradford, Clayton Kunz, Ron Kohavi, Cliff Brunk, and Carla E Brodley. Pruning decision trees with misclassification costs. In *European Conference on Machine Learning*, pages 131–136. Springer, 1998.

Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

Paula Branco. Re-sampling approaches for regression tasks under imbalanced domains. Master's thesis, Dep. Computer Science, Faculty of Sciences - University of Porto, 2014.

Paula Branco, Luis Torgo, and Rita P Ribeiro. A survey of predictive modelling under imbalanced distributions. *arXiv preprint arXiv:1505.01658*, 2015.

Paula Branco, Rita P Ribeiro, and Luis Torgo. UBL: an R package for utility-based learning. *arXiv preprint arXiv:1604.08079*, 2016a.

Paula Branco, Luís Torgo, and Rita P Ribeiro. A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys (CSUR)*, 49(2):31, 2016b.

Paula Branco, Luís Torgo, and Rita P Ribeiro. Exploring resampling with neighborhood bias on imbalanced regression problems. In *Portuguese Conference on Artificial Intelligence*, pages 513–524. Springer, 2017a.

Paula Branco, Luís Torgo, and Rita P Ribeiro. Relevance-based evaluation metrics for multi-class imbalanced domains. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 698–710. Springer, 2017b.

Paula Branco, Luís Torgo, and Rita P Ribeiro. SMOGN: a pre-processing approach for imbalanced regression. In *First International Workshop on Learning with Imbalanced Domains: Theory and Applications*, pages 36–50, 2017c.

Paula Branco, Luís Torgo, Rita P Ribeiro, Eibe Frank, Bernhard Pfahringer, and Markus Michael Rau. Learning through utility optimization in regression tasks. In *Data Science and Advanced Analytics (DSAA), 2017 IEEE International Conference on*, pages 30–39. IEEE, 2017d.

Paula Branco, Luís Torgo, and Rita P Ribeiro. MetaUtil: Meta learning for utility maximization in regression. In *International Conference on Discovery Science (to appear)*. Springer, 2018a.

Paula Branco, Luís Torgo, and Rita P Ribeiro. Resampling with neighbourhood bias on imbalanced domains. *Expert Systems*, 2018b.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. Classification and regression trees. wadsworth & brooks. *Monterey, CA*, 1984.

Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. The balanced accuracy and its posterior distribution. In *Pattern recognition (ICPR), 2010 20th international conference on*, pages 3121–3124. IEEE, 2010.

R. Brüggemann, P. B Sørensen, D. Lerche, and L. Carlsen. Estimation of averaged ranks by a local partial order model#. *Journal of chemical information and computer sciences*, 44 (2):618–625, 2004.

Chumphol Bunkhumpornpat and Sitthichoke Subpaiboonkit. Safe level graph for synthetic minority over-sampling techniques. In *Communications and Information Technologies (ISCIT), 2013 13th International Symposium on*, pages 570–575. IEEE, 2013.

Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Advances in Knowledge Discovery and Data Mining*, pages 475–482. Springer, 2009.

Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. Mute: Majority under-sampling technique. In *Information, Communications and Signal Processing (ICICS) 2011 8th International Conference on*, pages 1–4. IEEE, 2011.

Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. Dbsmote: Density-based synthetic minority over-sampling technique. *Applied Intelligence*, 36(3): 664–684, 2012.

Michael C Burl, Lars Asker, Padhraic Smyth, Usama Fayyad, Pietro Perona, Larry Crumpler, and Jayne Aubele. Learning to recognize volcanoes on venus. *Machine Learning*, 30(2):165–194, 1998.

Michael Cain and Christian Janssen. Real estate price prediction under asymmetric loss. *Annals of the Institute of Statistical Mathematics*, 47(3):401–414, 1995.

Peng Cao, Dazhe Zhao, and Osmar R Zaïane. A pso-based cost-sensitive neural network for imbalanced data classification. In *Trends and Applications in Knowledge Discovery and Data Mining*, pages 452–463. Springer, 2013.

Cristiano Leite Castro and Antônio de Pádua Braga. Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data. *IEEE Trans. Neural Netw. Learning Syst.*, 24(6):888–899, 2013.

Edward Y Chang, Beitao Li, Gang Wu, and Kingshy Goh. Statistical learning for effective visual information retrieval. In *ICIP (3)*, pages 609–612, 2003.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *JAIR*, 16:321–357, 2002.

Nitesh V. Chawla, Aleksandar Lazarevic, Lawrence O. Hall, and Kevin W. Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In *Knowledge Discovery in Databases: PKDD 2003*, pages 107–119. Springer, 2003.

Nitesh V. Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations Newsletter*, 6(1):1–6, 2004.

Nitesh V. Chawla, Lawrence O. Hall, and Ajay Joshi. Wrapper-based computation and evaluation of sampling methods for imbalanced datasets. In *Proceedings of the 1st international workshop on Utility-based data mining*, pages 24–33. ACM, 2005.

Nitesh V. Chawla, David A. Cieslak, Lawrence O. Hall, and Ajay Joshi. Automatically countering imbalance and its empirical relationship to cost. *Data Mining and Knowledge Discovery*, 17(2):225–252, 2008.

Chao Chen, Andy Liaw, and Leo Breiman. Using random forest to learn imbalanced data. *University of California, Berkeley*, 2004.

Sheng Chen, Haibo He, and Edwardo A. Garcia. Ramoboost: Ranked minority oversampling in boosting. *Neural Networks, IEEE Transactions on*, 21(10):1624–1642, 2010.

Peter F. Christoffersen and Francis X. Diebold. Further results on forecasting and model selection under asymmetric loss. *Journal of applied econometrics*, 11(5):561–571, 1996.

Peter F. Christoffersen and Francis X. Diebold. Optimal prediction under asymmetric loss. *Econometric theory*, 13(06):808–817, 1997.

Yu-Meei Chyi. Classification analysis techniques for skewed class distribution problems. *Master Thesis, Department of Information Management, National Sun Yat-Sen University*, 2003.

David A. Cieslak and Nitesh V. Chawla. Learning decision trees for unbalanced data. In *Machine Learning and Knowledge Discovery in Databases*, pages 241–256. Springer, 2008.

David A. Cieslak, Thomas R. Hoens, Nitesh V. Chawla, and W Philip Kegelmeyer. Hellinger distance decision trees are robust and skew-insensitive. *Data Mining and Knowledge Discovery*, 24(1):136–158, 2012.

Gilles Cohen, Mélanie Hilario, Hugo Sax, Stéphane Hugonnet, and Antoine Geissbuhler. Learning from imbalanced data in surveillance of nosocomial infection. *Artificial Intelligence in Medicine*, 37(1):7–18, 2006.

Noel AC Cressie. Statistics for spatial data: Wiley series in probability and mathematical statistics. *Find this article online*, 1993.

Sven F. Crone, Stefan Lessmann, and Robert Stahlbock. Utility based data mining for time series analysis: cost-sensitive learning for neural network predictors. In *Proceedings of the 1st international workshop on Utility-based data mining*, pages 59–68. ACM, 2005.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *ICML'06: Proc. of the 23rd Int. Conf. on Machine Learning*, ACM ICPS, pages 233–240. ACM, 2006.

Philip J Davis and Philip Rabinowitz. *Methods of numerical integration*. Courier Corporation, 2007.

María Dolores Del Castillo and José Ignacio Serrano. A multistrategy approach for digital text categorization from imbalanced documents. *ACM SIGKDD Explorations Newsletter*, 6(1):70–79, 2004.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, and A. Weingessel. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2011.

Pedro Domingos. Knowledge acquisition from examples via multiple models. In *Machine Learning - International Workshop Then Conference* -, pages 98–106. Morgan Kaufmann Publishers, INC., 1997.

Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. In *KDD'99: Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*, pages 155–164. ACM Press, 1999.

John Doucette and Malcolm I. Heywood. Gp classification under imbalanced data sets: Active sub-sampling and auc approximation. In *Genetic Programming*, pages 266–277. Springer, 2008.

Dennis J. Drown, Taghi M. Khoshgoftaar, and Naeem Seliya. Evolutionary sampling and software quality modeling of high-assurance systems. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 39(5):1097–1107, 2009.

Chris Drummond and Robert C. Holte. Explicitly representing expected cost: an alternative to roc representation. In *KDD'00: Proc. of the 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 198–207. ACM, 2000a.

Chris Drummond and Robert C Holte. Explicitly representing expected cost: An alternative to roc representation. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 198–207. ACM, 2000b.

Chris Drummond and Robert C. Holte. Exploiting the cost (in) sensitivity of decision tree splitting criteria. In *ICML*, pages 239–246, 2000c.

Chris Drummond and Robert C. Holte. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on Learning from Imbalanced Datasets II*, volume 11. Citeseer, 2003.

Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

James P. Egan. Signal detection theory and {ROC} analysis. 1975.

Charles Elkan. The foundations of cost-sensitive learning. In *IJCAI'01: Proc. of 17th Int. Joint Conf. of Artificial Intelligence*, volume 1, pages 973–978. Morgan Kaufmann Publishers, 2001.

Şeyda Ertekin. Adaptive oversampling for imbalanced data classification. In *Information Sciences and Systems 2013*, pages 261–269. Springer, 2013.

Şeyda Ertekin, Jian Huang, Leon Bottou, and Lee Giles. Learning on the border: active learning in imbalanced data classification. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 127–136. ACM, 2007a.

Şeyda Ertekin, Jian Huang, and C Lee Giles. Active learning for class imbalance problem. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 823–824. ACM, 2007b.

Andrew Estabrooks and Nathalie Japkowicz. A mixture-of-experts framework for learning from imbalanced data sets. In *Advances in Intelligent Data Analysis*, pages 34–43. Springer, 2001.

Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1):18–36, 2004.

Wei Fan, Salvatore J. Stolfo, Junxin Zhang, and Philip K. Chan. Adacost: misclassification cost-sensitive boosting. In *ICML*, pages 97–105. Citeseer, 1999.

Tom Fawcett. Roc graphs with instance-varying costs. *Pattern Recognition Letters*, 27(8): 882–891, 2006a.

Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006b.

Tom Fawcett and Foster Provost. Activity monitoring: Noticing interesting changes in behavior. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 53–62. ACM, 1999.

Alberto Fernández, Salvador García, María José del Jesus, and Francisco Herrera. A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets. *Fuzzy Sets and Systems*, 159(18):2378–2398, 2008.

Alberto Fernández, María José del Jesus, and Francisco Herrera. On the 2-tuples based genetic tuning performance for fuzzy rule based classification systems in imbalanced data-sets. *Information Sciences*, 180(8):1268–1291, 2010.

César Ferri, Peter Flach, and José Hernández-Orallo. Learning decision trees using the area under the roc curve. In *ICML*, volume 2, pages 139–146, 2002.

César Ferri, Peter Flach, José Hernández-Orallo, and Athmane Senad. Modifying roc curves to incorporate predicted probabilities. In *Proceedings of the second workshop on ROC analysis in machine learning*, pages 33–40, 2005.

César Ferri, José Hernández-Orallo, and R Modroiu. An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27–38, 2009.

César Ferri, José Hernández-Orallo, and Peter A Flach. Brier curves: a new cost-based visualisation of classifier performance. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 585–592, 2011a.

César Ferri, José Hernández-Orallo, and Peter A Flach. A coherent interpretation of auc as a measure of aggregated classification performance. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 657–664, 2011b.

Andrew O. Finley and Sudipto Banerjee. *MBA: Multilevel B-spline Approximation*, 2014. URL `https://CRAN.R-project.org/package=MBA`. R package version 0.0-8.

Eibe Frank and Remco R Bouckaert. Conditional density estimation with class probability estimators. In *Asian Conference on Machine Learning*, pages 65–81. Springer, 2009.

Yoav Freund. Boosting a weak learning algorithm by majority. *Information and computation*, 121(2):256–285, 1995.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

Giorgio Fumera and Fabio Roli. Cost-sensitive learning in support vector machines. *VIII Convegno Associazione Italiana per L'Intelligenza Artificiale*, 2002.

Mikel Galar, Alberto Fernández, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(4):463–484, 2012.

Mikel Galar, Alberto Fernández, Edurne Barrenechea, and Francisco Herrera. Eusboost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. *Pattern Recognition*, 2013.

Ming Gao, Xia Hong, Sheng Chen, Chris J Harris, and Emad Khalaf. Pdfos: Pdf estimation based over-sampling for imbalanced two-class problems. *Neurocomputing*, 138:248–259, 2014.

Joaquín García, Salvador Derrac, Isaac Triguero, Cristobal J Carmona, and Francisco Herrera. Evolutionary-based selection of generalized instances for imbalanced classification. *Knowledge-Based Systems*, 25(1):3–12, 2012.

Salvador García and Francisco Herrera. Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evolutionary Computation*, 17(3):275–306, 2009.

Salvador García, José Ramón Cano, Alberto Fernández, and Francisco Herrera. A proposal of evolutionary prototype selection for class imbalance problems. In *Intelligent Data Engineering and Automated Learning–IDEAL 2006*, pages 1415–1423. Springer, 2006.

Vicente García, Ramón Alberto Mollineda, and José Salvador Sánchez. A new performance evaluation method for two-class imbalanced problems. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 917–925. Springer, 2008.

Vicente García, Ramón Alberto Mollineda, and José Salvador Sánchez. Index of balanced accuracy: A performance measure for skewed class distributions. In *Pattern Recognition and Image Analysis*, pages 441–448. Springer, 2009.

Vicente García, Ramón Alberto Mollineda, and José Salvador Sánchez. Theoretical analysis of a performance measure for imbalanced data. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 617–620. IEEE, 2010.

Peter Geibel, Ulf Brefeld, and Fritz Wysotzki. Perceptron and svm learning with generalized cost models. *Intelligent Data Analysis*, 8(5):439–455, 2004.

Alireza Ghasemi, Mohammad T Manzuri, Hamid R Rabiee, Mohammad H Rohban, and Siavash Haghiri. Active one-class learning by kernel density estimation. In *Machine Learning for Signal Processing (MLSP), 2011 IEEE International Workshop on*, pages 1–6. IEEE, 2011a.

Alireza Ghasemi, Hamid R Rabiee, Mohsen Fadaee, Mohammad T Manzuri, and Mohammad H Rohban. Active learning from positive and unlabeled data. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 244–250. IEEE, 2011b.

Clive W. Granger. Outline of forecast theory using generalized cost functions. *Spanish Economic Review*, 1(2):161–173, 1999.

Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In *Advances in intelligent computing*, pages 878–887. Springer, 2005.

David J. Hand. Measuring classifier performance: a coherent alternative to the area under the roc curve. *Machine learning*, 77(1):103–123, 2009.

Peter. E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:515–516, 1968.

Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1322–1328. IEEE, 2008.

José Hernández-Orallo. Soft (gaussian cde) regression models and loss functions. *arXiv preprint arXiv:1211.1043*, 2012.

José Hernández-Orallo. {ROC} curves for regression. *Pattern Recognition*, 46(12):3395 – 3411, 2013. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2013.06.014. URL `http://www.sciencedirect.com/science/article/pii/S0031320313002665`.

José Hernández-Orallo. Probabilistic reframing for cost-sensitive regression. *ACM Trans. Knowl. Discov. Data*, 8(4):17:1–17:55, August 2014. ISSN 1556-4681. doi: 10.1145/2641758. URL `http://doi.acm.org/10.1145/2641758`.

José Hernández-Orallo, Peter Flach, and César Ferri. A unified view of performance metrics: Translating threshold choice into expected classification loss. *The Journal of Machine Learning Research*, 13(1):2813–2869, 2012.

José Hernández-Orallo, Adolfo Martínez-Usó, Ricardo BC Prudêncio, Meelis Kull, Peter Flach, Chowdhury Farhan Ahmed, and Nicolas Lachiche. Reframing in context: A systematic approach for model reuse in machine learning. *AI Communications*, 29(5): 551–566, 2016.

P.H. Hiemstra, E.J. Pebesma, C.J.W. Twenhöfel, and G.B.M. Heuvelink. Real-time automatic interpolation of ambient gamma dose rates from the dutch radioactivity monitoring network. *Computers & Geosciences*, 2008. DOI: http://dx.doi.org/10.1016/j.cageo.2008.10.011.

Robert C. Holte, Liane E. Acker, and Bruce W. Porter. Concept learning and the problem of small disjuncts. In *IJCAI*, volume 89, pages 813–818. Citeseer, 1989.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Shengguo Hu, Yanfeng Liang, Lintao Ma, and Ying He. Msmote: improving classification performance when training data is imbalanced. In *Computer Science and Engineering, 2009. WCSE'09. Second International Workshop on*, volume 2, pages 13–17. IEEE, 2009.

Kaizhu Huang, Haiqin Yang, Irwin King, and Michael R. Lyu. Learning classifiers from imbalanced data based on biased minimax probability machine. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–558. IEEE, 2004.

Jae Pil Hwang, Seongkeun Park, and Euntai Kim. A new weighted approach to imbalanced data classification problem via support vector machine with quadratic cost function. *Expert Systems with Applications*, 38(7):8580–8585, 2011.

Tasadduq Imam, Kai Ming Ting, and Joarder Kamruzzaman. z-svm: An svm for improved classification of imbalanced data. In *AI 2006: Advances in Artificial Intelligence*, pages 264–273. Springer, 2006.

Natalie Japkowicz. Assessment metrics for imbalanced learning. In Haibo He and Yunqian Ma, editors, *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons, 2013.

Nathalie Japkowicz. Learning from imbalanced data sets: a comparison of various strategies. In *AAAI workshop on learning from imbalanced data sets*, volume 68. Menlo Park, CA, 2000a.

Nathalie Japkowicz. The class imbalance problem: Significance and strategies. In *Proc. of the Int'l Conf. on Artificial Intelligence*, 2000b.

Nathalie Japkowicz, Catherine Myers, and Mark Gluck. A novelty detection approach to classification. In *IJCAI*, pages 518–523, 1995.

Piyasak Jeatrakul, Kok Wai Wong, and Chun Che Fung. Classification of imbalanced data by combining the complementary neural network and smote algorithm. In *Neural Information Processing. Models and Applications*, pages 152–159. Springer, 2010.

Taeho Jo and Nathalie Japkowicz. Class imbalances versus small disjuncts. *ACM SIGKDD Explorations Newsletter*, 6(1):40–49, 2004.

Mahesh V. Joshi, Vipin Kumar, and Ramesh C. Agarwal. Evaluating boosting algorithms to classify rare classes: Comparison and improvements. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 257–264. IEEE, 2001.

Jaz S Kandola and John Shawe-Taylor. Refining kernels for regression and uneven classification problems. In *AISTATS*, 2003.

Pilsung Kang and Sungzoon Cho. Eus svms: Ensemble of under-sampled svms for data imbalance problems. In *Neural Information Processing*, pages 837–846. Springer, 2006.

Grigoris I Karakoulas and John Shawe-Taylor. Optimizing classifers for imbalanced training sets. In *Advances in neural information processing systems*, pages 253–259, 1999.

Taghi M Khoshgoftaar, Chris Seiffert, Jason Van Hulse, Amri Napolitano, and Andres Folleco. Learning with limited minority class data. In *Machine Learning and Applications, 2007. ICMLA 2007. Sixth International Conference on*, pages 348–353. IEEE, 2007.

Sotiris Kotsiantis and Panagiotis Pintelas. Mixture of expert agents for handling imbalanced data sets. *Annals of Mathematics, Computing & Teleinformatics*, 1(1):46–55, 2003.

B. Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, pages 1–12, 2016.

Marek Kretowski and Marek Grześ. Evolutionary induction of decision trees for misclassification cost minimization. In *International Conference on Adaptive and Natural Computing Algorithms*, pages 1–10. Springer, 2007.

Miroslav Kubat and Stan Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In *Proc. of the 14th Int. Conf. on Machine Learning*, pages 179–186. Morgan Kaufmann, 1997.

Miroslav Kubat, Robert C Holte, and Stan Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine learning*, 30(2-3):195–215, 1998.

Matjaz Kukar and Igor Kononenko. Cost-sensitive learning with neural networks. In *ECAI*, pages 445–449, 1998.

Jorma Laurikkala. *Improving identification of difficult small classes by balancing class distribution*. Springer, 2001.

Hyoung-joo Lee and Sungzoon Cho. The novelty detection approach for different degrees of class imbalance. In *Neural Information Processing*, pages 21–30. Springer, 2006.

Sauchi Stephen Lee. Regularization in skewed binary classification. *Computational Statistics*, 14(2):277, 1999.

Sauchi Stephen Lee. Noisy replication in skewed binary classification. *Computational statistics & data analysis*, 34(2):165–191, 2000.

Seungyong Lee, George Wolberg, and Sung Yong Shin. Scattered data interpolation with multilevel b-splines. *IEEE transactions on visualization and computer graphics*, 3(3):228–244, 1997.

Tae-Hwy Lee. Loss functions in time series forecasting. *International encyclopedia of the social sciences*, 2008.

Chen Li, Chen Jing, and Gao Xin-tao. An improved p-svm method used to deal with imbalanced data sets. In *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, volume 1, pages 118–122. IEEE, 2009.

Jin Li, Xiaoli Li, and Xin Yao. Cost-sensitive classification with genetic programming. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 3, pages 2114–2121. IEEE, 2005.

Kewen Li, Wenrong Zhang, Qinghua Lu, and Xianghua Fang. An improved smote imbalanced data classification method based on support degree. In *Identification, Information and Knowledge in the Internet of Things (IIKI), 2014 International Conference on*, pages 34–38. IEEE, 2014.

Peng Li, Pei-Li Qiao, and Yuan-Chao Liu. A hybrid re-sampling method for svm learning from imbalanced data sets. In *Fuzzy Systems and Knowledge Discovery, 2008. FSKD'08. Fifth International Conference on*, volume 2, pages 65–69. IEEE, 2008.

Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2 (3):18–22, 2002.

Pin Lim, Chi Keong Goh, and Kay Chen Tan. Evolutionary cluster-based synthetic oversampling ensemble (eco-ensemble) for imbalance learning. *IEEE transactions on cybernetics*, 47(9):2850–2861, 2017.

FY Lin and S McClean. The prediction of financial distress using a cost sensitive approach and prior probabilities. In *Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning (WCSL at ICML-2000)*, 2000.

Yi Lin, Yoonkyung Lee, and Grace Wahba. Support vector machines for classification in nonstandard situations. *Machine learning*, 46(1):191–202, 2002.

Charles X. Ling and Victor S. Sheng. Cost-sensitive learning. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 231–235. Springer US, Boston, MA, 2011a.

Charles X. Ling and Victor S. Sheng. Cost-sensitive learning and the class imbalance problem. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 167–168. Springer, Boston, MA, 2011b.

Charles X Ling, Qiang Yang, Jianning Wang, and Shichao Zhang. Decision trees with minimal costs. In *Proceedings of the twenty-first international conference on Machine learning*, page 69. ACM, 2004.

Charles X Ling, Victor S Sheng, Tilmann Bruckhaus, and Nazim H Madhavji. Maximum profit mining and its application in software development. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 929–934. ACM, 2006.

Alexander Liu, Joydeep Ghosh, and Cheryl E. Martin. Generative oversampling for mining imbalanced datasets. In *DMIN*, pages 66–72, 2007.

Wei Liu, Sanjay Chawla, David A. Cieslak, and Nitesh V. Chawla. A robust decision tree algorithm for imbalanced data sets. In *SDM*, volume 10, pages 766–777. SIAM, 2010.

Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(2):539–550, 2009.

Yang Liu, Aijun An, and Xiangji Huang. Boosting prediction accuracy on imbalanced datasets with svm ensembles. In *Advances in Knowledge Discovery and Data Mining*, pages 107–118. Springer, 2006.

Susan Lomax and Sunil Vadera. A survey of cost-sensitive decision tree induction algorithms. *ACM Computing Surveys (CSUR)*, 45(2):16, 2013.

Tomasz Maciejewski and Jerzy Stefanowski. Local neighbourhood extension of smote for mining imbalanced data. In *Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on*, pages 104–111. IEEE, 2011.

Satyam Maheshwari, Jitendra Agrawal, and Sanjeev Sharma. A new approach for classification of highly imbalanced datasets using evolutionary algorithms. *Intl. J. Sci. Eng. Res*, 2:1–5, 2011.

Marcus A Maloof. Learning when data sets are imbalanced and when costs are unequal and unknown. In *ICML-2003 workshop on learning from imbalanced data sets II*, volume 2, pages 2–1, 2003.

Larry Manevitz and Malik Yousef. One-class svms for document classification. *the Journal of machine Learning research*, 2:139–154, 2002.

Inderjeet Mani and Jianping Zhang. knn approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of Workshop on Learning from Imbalanced Datasets*, 2003.

Dragos Margineantu. Building ensembles of classifiers for loss minimization. *Computing Science and Statistics*, pages 190–194, 1999.

Dragos Margineantu. On class probability estimates and cost-sensitive evaluation of classifiers. In *Workshop on Cost-Sensitive Learning at the 17th International Conference on Machine Learning*, 2000.

Dragos D Margineantu. Class probability estimation and cost-sensitive classification decisions. In *ECML*, pages 270–281. Springer, 2002.

Dragos Dorin Margineantu. *Methods for cost-sensitive learning*. PhD thesis, 2001.

José Manuel Martínez-García, Carmen Paz Suárez-Araujo, and Patricio García Báez. Sneom: a sanger network based extended over-sampling method. application to imbalanced biomedical datasets. In *Neural Information Processing*, pages 584–592. Springer, 2012.

Hamed Masnadi-Shirazi and Nuno Vasconcelos. Asymmetric boosting. In *Proceedings of the 24th international conference on Machine learning*, pages 609–619. ACM, 2007.

Hamed Masnadi-Shirazi, Nuno Vasconcelos, and Arya Iranmehr. Cost-sensitive support vector machines. *arXiv preprint arXiv:1212.0975*, 2012.

David Mease, Abraham Wyner, and Andreas Buja. Cost-weighted boosting with jittering and over/under-sampling: Jous-boost. *J. Machine Learning Research*, 8:409–439, 2007.

Giovanna Menardi and Nicola Torelli. Training and assessing classification rules with imbalanced data. *Data Mining and Knowledge Discovery*, pages 1–31, 2010.

Charles E Metz. Basic principles of roc analysis. In *Seminars in nuclear medicine*, volume 8 No.4, pages 283–298. Elsevier, 1978.

Ying Mi. Imbalanced classification based on active learning smote. *Research Journal of Applied Sciences*, 5, 2013.

S. Milborrow. *earth: Multivariate Adaptive Regression Spline Models. Derived from mda:mars by Trevor Hastie and Rob Tibshirani.*, 2012.

Stephanie L Moret, William T Langford, and Dragos D Margineantu. Learning to predict channel stability using biogeomorphic features. *Ecological Modelling*, 191(1):47–57, 2006.

Douglas Mossman. Three-way rocs. *Medical Decision Making*, 19(1):78–89, 1999.

Satuluri Naganjaneyulu and Mrithyumjaya Rao Kuppa. A novel framework for class imbalance learning using intelligent under-sampling. *Progress in Artificial Intelligence*, 2(1):73–84, 2013.

Munehiro Nakamura, Yusuke Kajiwara, Atsushi Otsuka, and Haruhiko Kimura. Lvq-smote–learning vector quantization based synthetic minority over–sampling technique for biomedical data. *BioData mining*, 6(1):16, 2013.

Krystyna Napierała, Jerzy Stefanowski, and Szymon Wilk. Learning from imbalanced data in presence of noisy and borderline examples. In *Rough Sets and Current Trends in Computing*, pages 158–167. Springer, 2010.

Wing WY Ng, Jiankun Hu, Daniel S Yeung, Sha Yin, and Fabio Roli. Diversified sensitivity-based undersampling for imbalance classification problems. 2014.

Nir Ofek, Lior Rokach, Roni Stern, and Asaf Shabtai. Fast-cbus: A fast clustering-based undersampling method for addressing the class imbalance problem. *Neurocomputing*, 243: 88–102, 2017.

Sang-Hoon Oh. Error back-propagation algorithm for classification of imbalanced data. *Neurocomputing*, 74(6):1058–1061, 2011.

Adam Omielan and Sunil Vadera. Ecco: A new evolutionary classifier with cost optimisation. *Intelligent Information Processing VI*, pages 97–105, 2012.

Michael Pazzani, Christopher Merz, Patrick Murphy, Kamal Ali, Timothy Hume, and Clifford Brunk. Reducing misclassification costs. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 217–225, 1994.

Edzer J. Pebesma. Multivariable geostatistics in s: the gstat package. *Computers & Geosciences*, 30:683–691, 2004.

Clifton Phua, Damminda Alahakoon, and Vincent Lee. Minority report in fraud detection: classification of skewed data. *ACM SIGKDD Explorations Newsletter*, 6(1):50–59, 2004.

Foster Provost and Tom Fawcett. Robust classification for imprecise environments. *Machine learning*, 42(3):203–231, 2001.

Foster J Provost and Tom Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *KDD*, volume 97, pages 43–48, 1997.

Foster J Provost, Tom Fawcett, and Ron Kohavi. The case against accuracy estimation for comparing induction algorithms. In *ICML'98: Proc. of the 15th Int. Conf. on Machine Learning*, pages 445–453. Morgan Kaufmann Publishers, 1998.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018. URL `http://www.R-project.org/`.

Enislay Ramentol, Yailé Caballero, Rafael Bello, and Francisco Herrera. Smote-rsb*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using smote and rough sets theory. *Knowledge and Information Systems*, 33(2):245–265, 2012a.

Enislay Ramentol, Nelle Verbiest, Rafael Bello, Yailé Caballero, Chris Cornelis, and Francisco Herrera. Smote-frst: a new resampling method using fuzzy rough set theory. In *10th International FLINS conference on uncertainty modelling in knowledge engineering and decision making (to appear)*, 2012b.

Romesh Ranawana and Vasile Palade. Optimized precision-a new measure for classifier performance evaluation. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 2254–2261. IEEE, 2006.

Bhavani Raskutti and Adam Kowalczyk. Extreme re-balancing for svms: a case study. *ACM Sigkdd Explorations Newsletter*, 6(1):60–69, 2004.

Markus Michael Rau, Stella Seitz, Fabrice Brimioulle, Eibe Frank, Oliver Friedrich, Daniel Gruen, and Ben Hoyle. Accurate photometric redshift probability density estimation–method comparison and application. *Monthly Notices of the Royal Astronomical Society*, 452(4):3710–3725, 2015.

Rita P Ribeiro. *Utility-based Regression*. PhD thesis, Dep. Computer Science, Faculty of Sciences - University of Porto, 2011.

Rita P Ribeiro and Luís Torgo. Predicting harmful algae blooms. In *Progress in Artificial Intelligence*, pages 308–312. Springer, 2003.

Cornelis V. Rijsbergen. Information retrieval. dept. of computer science, university of glasgow, 2nd edition. 1979.

Juan J Rodríguez, José-Francisco Díez-Pastor, Jesús Maudes, and César García-Osorio. Disturbing neighbors ensembles of trees for imbalanced data. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 83–88. IEEE, 2012.

José A Sáez, Julián Luengo, Jerzy Stefanowski, and Francisco Herrera. Smote–ipf: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. *Information Sciences*, 291:184–203, 2015.

Juan Pablo Sánchez-Crisostomo, Roberto Alejo, Erika López-González, Rosa María Valdovinos, and J Horacio Pacheco-Sánchez. Empirical analysis of assessments metrics for multi-class imbalance learning on the back-propagation context. In *Advances in Swarm Intelligence*, pages 17–23. Springer, 2014.

Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Rusboost: A hybrid approach to alleviating class imbalance. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 40(1):185–197, 2010.

Shiven Sharma, Colin Bellinger, and Nathalie Japkowicz. Clustering based one-class classification for compliance verification of the comprehensive nuclear-test-ban treaty. In *Advances in Artificial Intelligence*, pages 181–193. Springer, 2012.

Victor S Sheng and Charles X Ling. Thresholding for making classifiers cost-sensitive. In *AAAI*, pages 476–481, 2006.

Bernard W Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

Parinaz Sobhani, Herna Viktor, and Stan Matwin. Learning from imbalanced data using ensemble methods and cluster-based undersampling. In *New Frontiers in Mining Complex Patterns*, pages 69–83. Springer, 2014.

Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

Jie Song, Xiaoling Lu, and Xizhi Wu. An improved adaboost algorithm for unbalanced classification data. In *Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on*, volume 1, pages 109–113. IEEE, 2009.

Panote Songwattanasiri and Krung Sinapiromsaran. Smoute: Synthetics minority oversampling and under-sampling techniques for class imbalanced problem. In *Proceedings of the Annual International Conference on Computer Science Education: Innovation and Technology, Special Track: Knowledge Discovery*, pages 78–83, 2010.

Jerzy Stefanowski and Szymon Wilk. Selective pre-processing of imbalanced data for improving classification performance. In *Data Warehousing and Knowledge Discovery*, pages 283–292. Springer, 2008.

Yanmin Sun, Mohamed S Kamel, and Yang Wang. Boosting for learning multiple classes with imbalanced class distribution. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 592–602. IEEE, 2006.

Yanmin Sun, Mohamed S Kamel, Andrew KC Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

Aik Tan, David Gilbert, and Yves Deville. Multi-class protein fold classification using a new ensemble machine learning approach. 2003.

Yuchun Tang and Yan-Qing Zhang. Granular svm with repetitive undersampling for highly imbalanced protein homology prediction. In *Granular Computing, 2006 IEEE International Conference on*, pages 457–460. IEEE, 2006.

Yuchun Tang, Yan-Qing Zhang, Nitesh V. Chawla, and Sven Krasser. Svms modeling for highly imbalanced classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(1):281–288, 2009.

Dacheng Tao, Xiaoou Tang, Xuelong Li, and Xindong Wu. Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(7):1088–1099, 2006.

Nguyen Thai-Nghe, Zeno Gantner, and Lars Schmidt-Thieme. A new evaluation measure for learning from imbalanced data. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 537–542. IEEE, 2011.

Kai Ming Ting. Inducing cost-sensitive trees via instance weighting. In *European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 139–147. Springer, 1998.

Kai Ming Ting. A comparative study of cost-sensitive boosting algorithms. In *In Proceedings of the 17th International Conference on Machine Learning*. Citeseer, 2000a.

Kai Ming Ting. An empirical study of metacost using boosting algorithms. In *European Conference on Machine Learning*, pages 413–425. Springer, 2000b.

Kai Ming Ting. An instance-weighting method to induce cost-sensitive trees. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):659–665, 2002.

Kai Ming Ting and Zijian Zheng. Boosting trees for cost-sensitive classifications. In *European Conference on Machine Learning*, pages 190–195. Springer, 1998.

L. Torgo. An infra-structure for performance estimation and experimental comparison of predictive models in R. *CoRR*, abs/1412.0436, 2014.

Luís Torgo. Regression error characteristic surfaces. In *KDD'05: Proc. of the 11th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 697–702. ACM Press, 2005.

Luis Torgo. *Data mining with R: learning with case studies*. CRC press, Boca Raton, New York, UK, 2016.

Luís Torgo and Rita P Ribeiro. Predicting outliers. In *Knowledge Discovery in Databases: PKDD 2003*, pages 447–458. Springer, 2003.

Luís Torgo and Rita P Ribeiro. Utility-based regression. In *PKDD'07: Proc. of 11th European Conf. on Principles and Practice of Knowledge Discovery in Databases*, pages 597–604. Springer, 2007.

Luís Torgo and Rita P Ribeiro. Precision and recall in regression. In *DS'09: 12th Int. Conf. on Discovery Science*, pages 332–346. Springer, 2009.

Luís Torgo, Rita P Ribeiro, Bernhard Pfahringer, and Paula Branco. Smote for regression. In *Progress in Artificial Intelligence*, pages 378–389. Springer, 2013.

Luís Torgo, Paula Branco, Rita P Ribeiro, and Bernhard Pfahringer. Resampling strategies for regression. *Expert Systems*, 32(3):465–476, 2015.

Peter D Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of artificial intelligence research*, 2:369–409, 1995.

Peter D. Turney. Types of cost in inductive concept learning. *CoRR*, cs.LG/0212034, 2002. URL `http://arxiv.org/abs/cs.LG/0212034`.

Sunil Vadera. Csnl: A cost-sensitive non-linear decision tree algorithm. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(2):6, 2010.

Jason Van Hulse, Taghi M Khoshgoftaar, and Amri Napolitano. Experimental perspectives on learning from imbalanced data. In *Proceedings of the 24th international conference on Machine learning*, pages 935–942. ACM, 2007.

Madireddi Vasu and Vadlamani Ravi. A hybrid under-sampling approach for mining unbalanced datasets: applications to banking and insurance. *International Journal of Data Mining, Modelling and Management*, 3(1):75–105, 2011.

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. URL `http://www.stats.ox.ac.uk/pub/MASS4`.

Nele Verbiest, Enislay Ramentol, Chris Cornelis, and Francisco Herrera. Improving smote with fuzzy rough prototype selection to detect noise in imbalanced classification data. In *Advances in Artificial Intelligence–IBERAMIA 2012*, pages 169–178. Springer, 2012.

E. Grosse W. S. Cleveland and W. M. Shyu. Local regression models. In John M Chambers and Trevor J Hastie, editors, *Statistical models in S*, chapter 8. CRC Press, Inc., 1991.

Kiri L Wagstaff, Nina L Lanza, David R Thompson, Thomas G Dietterich, and Martha S Gilmore. Guiding scientific discovery with explanations using demud. In *AAAI*, 2013.

Byron C Wallace, Kevin Small, Carla E Brodley, and Thomas A Trikalinos. Class imbalance, redux. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 754–763. IEEE, 2011.

Chunru Wan, Lipo Wang, and Kai Ming Ting. Introducing cost-sensitive neural networks. In *Proceedings of the 2nd International Conference on Information, Communications and Signal Processing, Singapore*, pages 445–449, 1999.

Benjamin X Wang and Nathalie Japkowicz. Boosting support vector machines for imbalanced data sets. *Knowledge and information systems*, 25(1):1–20, 2010.

He-Yong Wang. Combination approach of smote and biased-svm for imbalanced datasets. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 228–231. IEEE, 2008.

Shuo Wang and Xin Yao. Diversity analysis on imbalanced data sets by using ensemble models. In *Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on*, pages 324–331. IEEE, 2009.

Deng Weiguo, Wang Li, Wang Yiyang, and Qian Zhong. An improved svm-km model for imbalanced datasets. In *Industrial Control and Electronics Engineering (ICICEE), 2012 International Conference on*, pages 100–103. IEEE, 2012.

Gary Weiss, Maytal Saar-Tsechansky, and Bianca Zadrozny (editors). UBDM'05: Proceedings of the 1st international workshop on utility-based data mining. held in the context of the 11th ACM SIGKDD international conference on knowledge discovery and data mining (KDD'05). 2005.

Gary M Weiss. Mining with rarity: a unifying framework. *SIGKDD Explorations Newsletter*, 6(1):7–19, 2004.

Gary M Weiss. Foundations of imbalanced learning. In Haibo He and Yunqian Ma, editors, *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons, 2013.

Gary M Weiss and Foster J Provost. Learning when training data are costly: the effect of class distribution on tree induction. *J. Artif. Intell. Res.(JAIR)*, 19:315–354, 2003.

Cheng G Weng and Josiah Poon. A new evaluation measure for imbalanced datasets. In *Proceedings of the 7th Australasian Data Mining Conference-Volume 87*, pages 27–32. Australian Computer Society, Inc., 2008.

Gang Wu and Edward Y Chang. Adaptive feature-space conformal transformation for imbalanced-data learning. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 816–823, 2003a.

Gang Wu and Edward Y Chang. Class-boundary alignment for imbalanced dataset learning. In *ICML 2003 workshop on learning from imbalanced data sets II, Washington, DC*, pages 49–56, 2003b.

Gang Wu and Edward Y Chang. Kba: Kernel boundary alignment considering imbalanced data distribution. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):786–795, 2005.

Shaomin Wu, Peter Flach, and César Ferri. An improved model selection heuristic for auc. In *ECML*, pages 478–489. Springer, 2007.

Xiaoyun Wu and Rohini K Srihari. New\'{\i}-support vector machines and their sequential minimal optimization. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 824–831, 2003.

Jin Xiao, Ling Xie, Changzheng He, and Xiaoyi Jiang. Dynamic classifier ensemble model for customer classification with imbalanced class distribution. *Expert Systems with Applications*, 39(3):3668–3675, 2012.

Yaya Xie, Xiu Li, EWT Ngai, and Weiyun Ying. Customer churn prediction using improved balanced random forests. *Expert Systems with Applications*, 36(3):5445–5449, 2009.

Fan Yang, Hua-zhen Wang, Hong Mi, Wei-wen Cai, et al. Using random forest for reliable classification and cost-sensitive learning for medical diagnosis. *BMC bioinformatics*, 10 (1):S22, 2009.

Qiang Yang and Xindong Wu. 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, 5(04):597–604, 2006.

Zeping Yang and Daqi Gao. An active under-sampling approach for imbalanced data classification. In *Computational Intelligence and Design (ISCID), 2012 Fifth International Symposium on*, volume 2, pages 270–273. IEEE, 2012.

Show-Jane Yen and Yue-Shi Lee. Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset. In *Intelligent Control and Automation*, pages 731–740. Springer, 2006.

Show-Jane Yen and Yue-Shi Lee. Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Systems with Applications*, 36(3):5718–5727, 2009.

Yang Yong. The research of imbalanced data set of sample sampling method based on k-means cluster and genetic algorithm. *Energy Procedia*, 17:164–170, 2012.

Kihoon Yoon and Stephen Kwek. An unsupervised learning approach to resolving the data imbalanced issue in supervised learning problems in functional genomics. In *Hybrid Intelligent Systems, 2005. HIS'05. Fifth International Conference on*, pages 6–pp. IEEE, 2005.

Dai Yuanhong, Chen Hongchang, and Peng Tao. Cost-sensitive support vector machine based on weighted attribute. In *Information Technology and Applications, 2009. IFITA'09. International Forum on*, volume 1, pages 690–692. IEEE, 2009.

Bianca Zadrozny and Charles Elkan. Learning and making decisions when costs and probabilities are both unknown. In *KDD'01: Proceedings of the 7th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 204–213. ACM Press, 2001.

Bianca Zadrozny, John Langford, and Naoki Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 435–442. IEEE, 2003.

Bianca Zadrozny, Gary Weiss, and Maytal Saar-Tsechansky (editors). UBDM'06: Proceedings of the 2nd international workshop on utility-based data mining. held in the context of the 12th ACM SIGKDD international conference on knowledge discovery and data mining (KDD' 06). 2006.

Arnold Zellner. Bayesian estimation and prediction using asymmetric loss functions. *Journal of the American Statistical Association*, 81(394):446–451, 1986.

Dongmei Zhang, Wei Liu, Xiaosheng Gong, and Hui Jin. A novel improved smote resampling algorithm based on fractal. *Journal of Computational Information Systems*, 7(6):2204–2211, 2011.

Huaxiang Zhang and Mingfang Li. Rwo-sampling: A random walk over-sampling approach to imbalanced data classification. *Information Fusion*, 20:99–116, 2014.

Shichao Zhang, Xiaofeng Zhu, Jilian Zhang, and Chengqi Zhang. Cost-time sensitive decision tree with missing values. *Knowledge Science, Engineering and Management*, pages 447–459, 2007.

Huimin Zhao, Atish P Sinha, and Gaurav Bansal. An extended tuning method for cost-sensitive regression and forecasting. *Decision Support Systems*, 51(3):372–383, 2011.

Jun Zheng. Cost-sensitive boosting neural networks for software defect prediction. *Expert Systems with Applications*, 37(6):4537–4543, 2010.

Zhi-Hua Zhou and Xu-Ying Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *Knowledge and Data Engineering, IEEE Transactions on*, 18(1):63–77, 2006.

Zhi-Hua Zhou and Xu-Ying Liu. On multi-class cost-sensitive learning. *Computational Intelligence*, 26(3):232–257, 2010.

Jingbo Zhu and Eduard H Hovy. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. In *EMNLP-CoNLL*, volume 7, pages 783–790, 2007.

Ling Zhuang and Honghua Dai. Parameter optimization of kernel-based one-class classifier on imbalance learning. *Journal of Computers*, 1(7):32–40, 2006a.

Ling Zhuang and Honghua Dai. Parameter estimation of one-class svm on imbalance text classification. In *Advances in Artificial Intelligence*, pages 538–549. Springer, 2006b.