Pós-Graduação em Ciência da Computação

ROMERO FERNANDO ALMEIDA BARATA DE MORAIS

NEW SAMPLING ALGORITHMS FOR ENHANCING CLASSIFIER PERFORMANCE ON IMBALANCED DATA PROBLEMS

Federal University of Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2018

Romero Fernando Almeida Barata de Morais

NEW SAMPLING ALGORITHMS FOR ENHANCING CLASSIFIER PERFORMANCE ON IMBALANCED DATA PROBLEMS

A M.Sc. Dissertation presented to the Centre for Informatics of Federal University of Pernambuco in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Advisor: Germano Crispim Vasconcelos

RECIFE
2018

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

M827n Morais, Romero Fernando Almeida Barata de
New sampling algorithms for enhancing classifier performance on imbalanced data problems / Romero Fernando Almeida Barata de Morais. – 2018.
94 f.: il., fig., tab.

Orientador: Germano Crispim Vasconcelos.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2018.
Inclui referências.

1. Inteligência computacional. 2. Aprendizagem de máquina. I. Vasconcelos, Germano Crispim (orientador). II. Título.

006.3 CDD (23. ed.) UFPE- MEI 2018-045

Romero Fernando Almeida Barata de Morais

New Sampling Algorithms for Enhancing Classifier Performance on Imbalanced Data Problems

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 06/02/2018

BANCA EXAMINADORA

_____

Profa. Dra. Patricia Cabral de Azevedo Restelli Tedesco
Centro de Informática / UFPE

_____

Prof. Alexandre Magno Andrade Maciel
Escola Politécnica de Pernambuco / UPE

_____

Prof. Dr. Germano Crispim Vasconcelos
Centro de Informática
(Orientador)

I dedicate this dissertation to all family, friends, professors, and my partner, who gave me the necessary support to conclude it.

Acknowledgements

Firstly, I would like to thank my family for supporting me whenever necessary, doing their best to assist me with anything required for my education.

A special thanks goes to professor Germano, for all his patience and dedication during these two years of Master's work, and for the many opportunities to work and learn together.

Friends were an important part during these two years and I could not forget to mention César and Péricles for their support and many insightful discussions.

The research I started at Motorola during the last year of my undergraduate degree continued for the first year of my Master's degree. I would like to say thank you again to professor Ricardo for the great opportunity to work in this research project.

Even though we interacted a lot less in these last two years, professors Silvio and Pedro were important mentors during my undergraduate years and inspired me to pursue an academic career. For them, my big thank you.

Last but not least, I would like to say thank you to my partner, Natalie, for her patience and love, and for bearing with me despite all shortcomings of living in different countries.

Read not to contradict and confute; nor to believe and take for granted; nor to find talk and discourse; but to weigh and consider.

—FRANCIS BACON

Abstract

Classification problems where the distribution of examples among the classes are imbalanced arise frequently in real-world domains. Commonly, these domains comprise critical problems where accurate predictions for all classes are necessary, such as credit card fraud detection, churn prediction, disease diagnosis, and network intrusive traffic detection. The problem with imbalanced data sets is that standard classifiers often have low accuracy on the underrepresented classes of the problem. Data sampling is the most popular approach to deal with imbalanced data sets and works by either decreasing the size of majority classes (under-sampling) or increasing the size of minority classes (over-sampling). In this dissertation we propose two new data sampling algorithms: RRUS and *k*-INOS. RRUS is an under-sampling algorithm that aims to select a subset of examples from the majority class that best represents the majority class by preserving its density distribution. *k*-INOS is a general strategy to enhance robustness of over-sampling algorithms to noisy examples present in the minority class. Both algorithms were extensively tested on 50 imbalanced data sets, 6 diverse classifiers, and performance was evaluated according to 7 metrics. In particular, RRUS was compared to other 3 under-sampling algorithms and was significantly better than KMUS and SBC most of the time, and significantly better than RUS many times, for most classifiers and performance metrics. *k*-INOS, as a wrapper for over-sampling algorithms, was tested on 7 over-sampling algorithms and significantly increased Accuracy, Precision, and Specificity most of the time, and F_1 many times. In addition, *k*-INOS' hyperparameters were studied and appropriate values for their use were suggested. Finally, rules extracted from the former experiments with *k*-INOS revealed that the N3 complexity metric (loocv error rate of the 1-NN classifier) is often an indicator of whether *k*-INOS is likely to attain performance improvements or not.

Keywords: Imbalanced Learning. Over-Sampling. Under-Sampling.

Resumo

Problemas de classificação onde a distribuição de exemplos entre as classes é desbalanceada advém frequentemente de problemas reais. Muitas vezes, tais problemas reais são de natureza crítica e predições corretas para exemplos de todas as classes são necessárias, como em detecção de fraudes em cartões de crédito, identificação de doenças raras, e detecção de tráfego intrusivo em redes de internet. A problemática associada a dados desbalanceados é que classificadores comuns tendem a ter uma baixa taxa de acerto nas classes minoritárias. Algoritmos de amostragem são a solução mais comum para reduzir o desbalanceamento e em geral diminuem o número de exemplos nas classes majoritárias (sub-amostragem) ou aumentam o número de exemplos nas classes minoritárias (super-amostragem). Nesta dissertação propomos dois novos algoritmos de amostragem: RRUS e *k*-INOS. RRUS é um algoritmo de sub-amostragem que tem como objetivo obter um subconjunto da classe majoritária que melhor representa a classe majoritária original, através da preservação da distribuição de densidade. *k*-INOS é uma estratégia que torna qualquer algoritmo de super-amostragem mais robusto a ruídos presentes na classe minoritária. Ambos os algoritmos foram extensivamente testados em 50 conjuntos de dados desbalanceados, 6 classificadores diversos, e a performance foi avaliada de acordo com 7 métricas. Em particular, RRUS foi comparado com outros 3 algoritmos de sub-amostragem e teve um desempenho significativamente melhor que KMUS e SBC na maioria das vezes, e significativamente melhor que RUS várias vezes, para a maioria dos classificadores e métricas de performance. *k*-INOS, por ser aplicável a qualquer algoritmo de super-amostragem, foi testado em 7 algoritmos de super-amostragem e melhorou de maneira significativa na maioria das vezes a taxa de acerto, a precisão, e a cobertura da classe majoritária, e melhorou de maneira significativa em vários casos a métrica F_1. Adicionalmente, os hyperparâmetros de *k*-INOS foram analisados através de um estudo de caso e valores apropriados para seu uso foram sugeridos. Por fim, um conjunto de regras foram extraídas a partir dos resultados principais com *k*-INOS e revelaram que a métrica de complexidade N3 (taxa de erro do 1-NN usando loocv) é frequentemente um indicador de situações em que *k*-INOS tem ou não chances de melhorar a performance de algoritmos de super-amostragem.

Palavras-chave: Aprendizagem Desbalanceada. Super-Amostragem. Sub-Amostragem.

List of Figures

List of Tables

List of Acronyms

ADASYN Adaptive Synthetic Sampling 19

AE Autoencoder 34

BDL-SMOTE Borderline-SMOTE 19

CBO Cluster-Based Over-Sampling 25

CNN Condensed Nearest Neighbour Rule 29

DBFS Density Based Feature Selection 24

DBMUTE Density-Based Majority Under-Sampling Technique 18

DBSCAN Density-Based Spatial Clustering of Applications with Noise 18

DBSMOTE Density-Based Synthetic Minority Over-sampling TEchnique 32

DSUS Diversified Sensitivity Based Under-Sampling 18

DT Decision Tree 17

ENN Wilson's Edited Nearest Neighbour Rule 19

GBM Gradient Boosting Machine 37

IPF Iterative-Partitioning Filter 33

***k*-INOS** *k*-Influential Neighbourhood for Over-Sampling 60

KMUS *k*-Means Under-Sampling 18

***k*-NN** *k*-Nearest Neighbours 37

LR Logistic Regression 37

MD Most Distant 30

NCL Neighbourhood Cleaning Rule 30

NM1 NearMiss-1 30

NM2 NearMiss-2 30

NM3 NearMiss-3 30

NRAS Noise Reduction A Priori Synthetic Over-Sampling 33

OSS One-Sided Selection 29

PCA Principal Component Analysis 34

PDFOS Probability Density Function Estimation Based Over-Sampling 33

PW Parzen Window 33

RACOG RApidly COnverging Gibbs Sampler 33

Contents

1

INTRODUCTION

1.1 Motivation

Learning from imbalanced data sets is an important area of research in machine learning as many real-world problems present an imbalance in the distribution of their examples. A few problems include *credit card fraud detection* (DAL POZZOLO et al., 2014), *software defect detection* (WANG; YAO, 2013), *diagnosis of diseases* (KRAWCZYK et al., 2016), and *churn prediction* (BUREZ; POEL, 2009). Common classifiers usually expect a balanced distribution of examples among classes of the problem, an assumption that if not met may hinder learning (HE; GARCIA, 2009). For instance, a Decision Tree (DT) classifier is trained by successively splitting the training data into smaller subsets based on individual features. If one or more classes are under-represented, the constructed tree may not induce strong rules for the minority class(es). Therefore, in order to appropriately learn about all classes of an imbalanced problem, the imbalance issue should not be ignored.

In addition, the difficulty in learning from imbalanced data sets does not stem only from the imbalance, but from the complexity of the data sets as well (LÓPEZ et al., 2013). Sources of complexity include noisy examples, overlapping between classes, multimodal density distributions, and lack of density in the data. Noisy examples tend to impact the learning process of most classifiers. However, in an imbalanced setting, the minority class is already under-represented and noisy examples present on the minority class account for a higher fraction of it (WEISS, 2004). Any linearly separable problem, regardless of the imbalance in it, can be accurately modelled by most classifiers. However, when the overlap between the classes is high, classifiers might be forced to learn a decision boundary that ignores the minority class (PRATI et al., 2004). A multimodal density distribution increases the difficulty in learning the target class as it splits the data into several clusters. If a minority class comes from a multimodal distribution, the small amount of examples available are split into clusters and can be easily confused with noise. Lack of density in the data refers to the small sample size and poses a problem as there is not enough information available to properly train a classifier (RAUDYS; JAIN et al., 1991).

Many techniques are available to aid classifiers in learning from imbalanced data sets. The most popular techniques are *data sampling*, *cost-sensitive learning*, and *ensemble learning*.

Data sampling techniques work at the data level by modifying the distribution of examples among the classes prior to training of a classifier. Commonly, *data sampling* techniques either increase the size of the minority class(es) (over-sampling) or decrease the size of the majority class(es) (under-sampling). On the other hand, *cost-sensitive learning* techniques work at the classifier level and modify their learning algorithms to take the imbalance into consideration (LING; SHENG, 2011). This is usually achieved by increasing the misclassification costs associated with examples from the minority class(es). Ensembles are known for their increased performance against individual classifiers and it is no surprise that their strengths are explored for the imbalanced learning problem. Usually, *ensemble learning* combine the modelling capabilities of ensembles with *data sampling* and *cost-sensitive learning* methods (GALAR et al., 2012). In this dissertation, we focus on *data sampling* techniques and do not investigate the other techniques further.

The simplest under-sampling algorithm is Random Under-Sampling (RUS). It is a non-heuristic algorithm that simply selects a random sample of examples from the majority class. Unless the sample size is big enough, RUS is unlikely to preserve the density distribution of the majority class (due to the law of large numbers). An adaptation of the k-Means algorithm leads to a common under-sampling algorithm called k-Means Under-Sampling (KMUS). It works by clustering the majority class with the k-Means algorithm and using the computed centroids as the representatives of the majority class. As k-Means tends to compute equal-sized clusters, the uniform distribution of the selected examples may not reflect the actual distribution of the majority examples. A further adaptation of the idea of clustering present in KMUS is also present in Under-Sampling Based on Clustering (SBC) (YEN; LEE, 2009), Diversified Sensitivity Based Under-Sampling (DSUS) (NG et al., 2015), and Density-Based Majority Under-Sampling Technique (DBMUTE) (BUNKHUMPORNPAT; SINAPIROMSARAN, 2017). SBC clusters both classes together and, for each cluster, randomly selects majority examples based on the imbalance ratio of the cluster. On the other hand, DSUS clusters majority and minority classes separately and repeatedly selects examples from both classes using a sensitivity measure coupled with a Radial Basis Function Neural Network (RBFNN). DBMUTE utilises the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) clustering algorithm and aims to remove examples in the overlapping region between the classes. Even though a more refined strategy to select majority examples are employed by these algorithms, their aim is not to preserve the density distribution of the majority class.

Over-sampling, on the other hand, increases the size of the minority class and its simplest algorithm is Random Over-Sampling (ROS). ROS is a non-heuristic algorithm that selects examples from the minority class at random and appends them to the data set. The main drawbacks of ROS are the fact that it does not add any new information to the data set and increases the chance of overfitting (HE; GARCIA, 2009). Synthetic Minority Over-Sampling TEchnique (SMOTE) (CHAWLA et al., 2002) was the first over-sampling algorithm to add new examples to the minority class and works by generating new examples in the line segment joining

two close examples in the minority class. SMOTE only looks at the examples from the minority class and may generate noisy examples due to this blindness. Based on the drawbacks of SMOTE, many adaptations to it were created, such as Borderline-SMOTE (BDL-SMOTE) (HAN; WANG; MAO, 2005), Adaptive Synthetic Sampling (ADASYN) (HE et al., 2008), and Safe-Level-SMOTE (SL-SMOTE) (BUNKHUMPORNPAT; SINAPIROMSARAN; LURSINSAP, 2009). All these variations synthesise new examples similarly to SMOTE, however they differ in the way they handle noisy examples. Another type of adaptation of the SMOTE algorithm was the inclusion of a cleaning step before or after over-sampling. A few examples include SMOTE + Wilson's Edited Nearest Neighbour Rule (ENN) and SMOTE + Tomek Link (TL). SMOTE + ENN and SMOTE + TL use, respectively, the ENN and TL algorithms to remove noisy examples, from both classes, already present in the data or potentially introduced by SMOTE. Even though all these adaptations of SMOTE manage to synthesise new examples, these new examples may not be genuine representatives of the minority class, i.e. they might not come from the underlying density distribution of the minority class.

The previous discussion about under-sampling and over-sampling algorithms show that current *data sampling* algorithms have several advantages and disadvantages, which leaves a lot of room for improvement. In particular for under-sampling algorithms, given under-sampled subsets of same size, are there subsets "better" than others? And for over-sampling algorithms, how does noisy examples affect over-sampling algorithms? These questions set the tone for this dissertation.

1.2 Research Objectives

The main objective of this dissertation is to further investigate the use of *data sampling* algorithms and propose two new methods to improve the performance of classifiers learning from imbalanced data sets.

To achieve this objective, two lines of investigation are pursued. The first one explores the benefits of preserving the density distribution of the majority class after under-sampling, while the second one analyses the impact of noise on over-sampling algorithms. To narrow down our investigation, we consider only imbalanced problems with two classes and numeric features.

1.3 Contributions

This dissertation makes the following major contributions:

1. *A new under-sampling algorithm that minimises the loss of information about the majority class and increases recognition of the minority class.*

   This dissertation proposes an under-sampling algorithm that minimises the loss of information by preserving the density distribution of the majority class. The algorithm

named Repeated Random Under-Sampling (RRUS), manages to keep a high level of accuracy for the majority class and increases the accuracy for the minority class.

2. *A general strategy to enhance the robustness of over-sampling algorithms to noisy examples from the minority class*.

   This dissertation proposes a general method to filter noisy examples from the minority class prior to over-sampling. The method is named *k*-INOS, short for *k*-Influential Neighbourhood for Over-Sampling, and is applicable to any over-sampling algorithm. *k*-INOS was tested on many over-sampling algorithms and improved their performance according to several metrics.

and the following minor contributions:

1. *Proposal of a suite of imbalanced data sets for scientific experimentation*.

   There is little consensus among researchers working on the imbalanced learning problem on which data sets to use for experimentation of new *data sampling* algorithms. In this dissertation, we propose the use of 50 imbalanced data sets selected according to a few criteria and collected from several sources.

2. *bimba: An R package that implements a variety of data sampling algorithms*.

   An R package implementing many data sampling algorithms was developed in order to assist the research of this dissertation. It is open-source and publicly available at https://github.com/RomeroBarata/bimba.

3. *dcme: An R package to compute data complexity measures*.

   An R package containing many functions to compute data complexity measures was developed in order to assist with some of the experiments in this dissertation. It is open-source and publicly available at https://github.com/RomeroBarata/dcme.

1.4 Structure of the Dissertation

The rest of this dissertation is organised as follows. Chapter 2 reviews the literature and is divided into two parts: a discussion of the problem of learning from imbalanced data sets and a review of the most relevant *data sampling* algorithms. The first part analyses the imbalanced learning problem, which includes the difficulties that arise when learning from imbalanced data sets, types and nature of imbalance, and best practices in evaluating classifiers learning from imbalanced data sets. The second part reviews the most important under-sampling and over-sampling algorithms, describing their behaviour and organising them according to their sampling strategy.

Chapter 3 reports the experimental methodology adopted in all experiments of this dissertation. This includes the suite of data sets utilised, the base classifiers, the performance metrics, and the training framework employed. Statistical tests, albeit used, were specific to each proposed method's experimentation and their discussions are delayed to the specific chapters.

Chapter 4 is about RRUS, the under-sampling algorithm proposed in this dissertation. The chapter starts by presenting an overview of the proposed method and reviewing works that are related to it. Then, a detailed description of RRUS is given, including examples and an algorithmic description of it. Next, the results of the experiments conducted are shown and discussed. Since RRUS is based on the RUS and KMUS under-sampling algorithms, a few individual comparisons are made and discussed between RRUS and RUS, and RRUS and KMUS. Finally, the chapter is summarised and the advantages and limitations of RRUS are highlighted

Chapters 5 and 6 are about k-INOS, the general strategy to enhance robustness of over-sampling algorithms to noise proposed in this dissertation. Chapter 5 has a structure similar to Chapter 4 and starts by giving an overview of k-INOS and reviewing related works. Next, the concept of neighbourhood of influence is explained, which is necessary to fully understand the mechanics of k-INOS, and the usefulness of k-INOS is demonstrated through a simple example. An algorithmic description of k-INOS is given as well. Then, specific details about k-INOS' experimentation are explained and the results are shown and discussed. Finally, the chapter is summarised and the benefits and drawbacks of using k-INOS are discussed.

Chapter 6 further analysis the k-INOS algorithm and is an extension of Chapter 5. It has two main sections: an independent study of k-INOS' hyperparameters and an analysis of the situations where k-INOS is most likely to attain improvements in performance. The first section uses the mammography data set as case study and conducts a grid search on a range of values for k-INOS' hyperparameters, using SMOTE as base over-sampling algorithm. The second section mines rules from the results presented in Chapter 5 to discover which characteristics are indicators of improvements or no improvements in performance by k-INOS.

Chapter 7 presents the conclusions of this dissertation, indicate several paths for future work, and discloses the articles that contain parts of this dissertation.

2

LITERATURE REVIEW

This chapter reviews the problem of learning from imbalanced data sets and describes the most important *data sampling* algorithms in literature.

2.1 Imbalanced Learning

In this section we discuss the problem of learning from imbalanced data sets. We start by defining what is an imbalanced data set and examining several problems that usually accompany imbalanced data sets, increasing the difficulty in learning from them. Next we discuss two types of imbalance that can happen in a data set: *between-class* and *within-class* imbalance. In addition to the types of imbalance, we discuss the nature of the imbalance as well. For instance, is the observed imbalance *intrinsic* or *extrinsic* to the problem? Finally, we highlight the fact that Accuracy is not an appropriate measure of performance to assess classifiers learning from imbalanced data sets and mention a number of performance metrics that are more appropriate to evaluate classifiers trained on imbalanced data sets.

2.1.1 What is Imbalanced Learning?

Any data set having an unequal distribution of examples among its classes, as in Figure 2.1, is an imbalanced data set. However, in practice, a data set is considered imbalanced if the number of examples in the majority class is much greater than the number of examples in the minority class. The name imbalanced learning, then, refers to any classifier learning from an imbalanced data set.

Common classifiers may struggle to correctly classify examples from the minority class, as their learning algorithm usually expect a balanced distribution of examples (WEISS; PROVOST, 2001). For instance, a rule-based classifier tries to learn rules that are simple yet broad enough to properly classify unseen examples. However, for imbalanced data sets, simple rules are more likely to cover more examples from majority class, which induces the rule-based classifier to classify most examples as belonging to the majority class. Nevertheless, the imbalance itself is not the only problem, since any linearly separable data set can have both

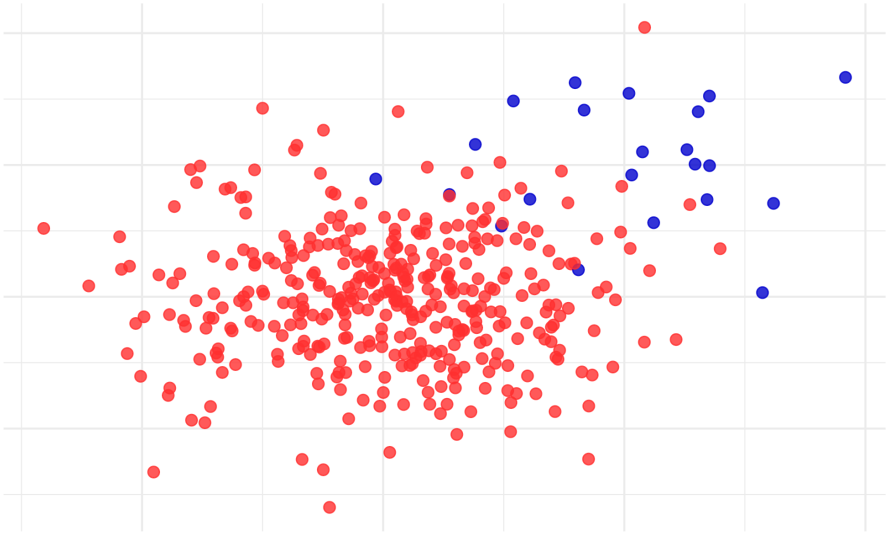

Figure 2.1: An imbalanced data set.

its classes correctly classified by most classifiers, regardless of the imbalance present in it. Many other complexity factors contribute to the difficulty in correctly classifying a data set and an imbalanced distribution of examples amplify the problems associated with these complexity factors (LÓPEZ et al., 2013).

Noise is a commonly known offender of classification performance in machine learning. In particular, for imbalanced data sets, as the minority class is already underrepresented, noisy examples make up for a larger proportion of the minority class and can severely impact learning (WEISS, 2004). A study conducted in (KHOSHGOFTAAR; VAN HULSE; NAPOLITANO, 2011) compared the performance of boosting and bagging techniques on noisy imbalanced data sets. Their experimentation used 4 data sets where the authors varied the imbalance ratio, the amount of noise injected, and the type of noise injected, to analyse the behaviour of 8 bagging/boosting techniques on noisy and imbalanced data sets. In addition, 4 base classifiers were employed and their performance evaluated on 7 performance metrics. They concluded that, for imbalanced data sets with little noise there is not much difference in performance between boosting and bagging algorithms, however, for increased levels of noise (specially minority examples mislabelled as majority examples) performance of bagging is far superior than performance of boosting. Thus, the authors recommend the use of bagging without replacement for noisy imbalanced data sets. In another work (SEIFFERT et al., 2014), the authors conducted an empirical study on the effects of noisy and imbalanced data sets on classifiers and *data sampling* algorithms. From a large software engineering data set, the authors derived 12 data sets with different levels of noise and imbalance. Then, 7 *data sampling* algorithms and 11 classifiers were tested on these data sets to assess the impact of noise and imbalance in learning. Performance was measured according to the AUROC metric. The authors concluded that most classifiers and *data sampling* algorithms are more sensitive to noise than imbalance in the data, unless the imbalance is severe.

Overlapping between the classes is another source of learning complexity and is characterised by having regions in feature space containing a similar amount of examples from distinct classes. A systematic study performed in (PRATI et al., 2004) on a set of artificial data sets analysed the impact of class overlap and class imbalance in learning. In total, 120 artificial data sets were generated by varying the imbalance ratio and degree of overlap between the classes, and they were used to train a C4.5 classifier which had its performance evaluated according to the AUROC metric. The authors concluded that class overlap hinders performance more than class imbalance. Following the same lead, (DENIL; TRAPPENBERG, 2010) studied the relationship between class overlap, data imbalance, and learned model complexity. Their findings agree with (PRATI et al., 2004), in which class overlap usually degrades performance more than the imbalance itself. In addition, they empirically showed that the negative effects produced by class overlap and data imbalance are not independent, as the degradation in performance does not follow what is predicted by a model (defined by the authors) that considers these factors independently. The complexity of the learned model, measured proportionally to the number of support vectors used by a trained Support Vector Machine (SVM), grows proportional to level of overlap between the classes and the size of the training data set, which has a counter-intuitive interpretation, that more training data might not be beneficial as it increases the complexity of the learned model.

When very few examples are available for the minority class, we have a small sample size problem (RAUDYS; JAIN et al., 1991), or lack of density in the data. In this case, the examples available do not provide enough information to allow for proper generalisation, and this problem is aggravated further if data are imbalanced and high-dimensional. The benefits of using feature selection on small high-dimensional imbalanced data sets were investigated by (WASIKOWSKI; CHEN, 2010). The authors studied the effect of 7 feature selection algorithms on 28 data sets spanning 3 different domains and used a linear SVM as base classifier evaluated by AUROC and AUPR performance metrics. Their main finding was that feature selection is most of the time beneficial when learning from small high-dimensional imbalanced data sets. A feature selection algorithm designed to deal with small high-dimensional imbalanced data sets was proposed by (ALIBEIGI; HASHEMI; HAMZEH, 2012). The proposed method, named Density Based Feature Selection (DBFS), works by ranking the features of a data set according to the probability density estimation of them for each class. DBFS was compared to other feature selection algorithms and attained the best results for majority of experiments performed by the authors, and supported the hypothesis that feature selection aids classifiers learning from small high-dimensional imbalanced data sets.

2.1.2 Types and Nature of Imbalance

The disparity between the number of examples in the majority class and the minority class is commonly deemed as the only source of imbalance in a data set. However, an individual

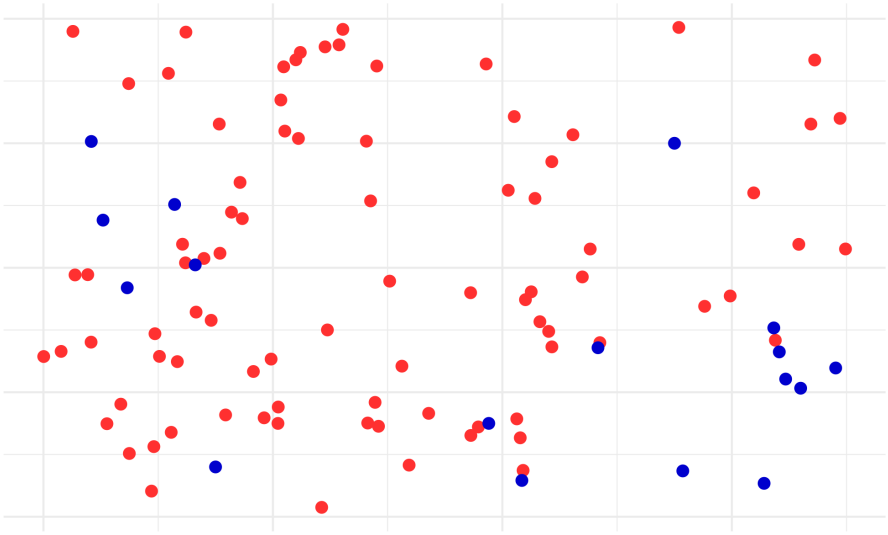

Figure 2.2: An imbalanced data set containing a *within-class* imbalance.

class may have a multimodal density distribution and several modes might be underrepresented. Therefore, not only we can have a *between-class* imbalance, but we can also have a *within-class* imbalance as depicted in Figure 2.2.

The problem of having a multimodal density distribution, with underrepresented modes, is closely related to the problem of small disjuncts (HE; GARCIA, 2009). And since articles rarely discuss the problem of underrepresented clusters from the point of view of a multimodal density distribution, we briefly review here the works that analyse the problem of small disjuncts.

Small disjuncts stem from rule-based and tree-based classifiers and are defined as rules that correctly classify few examples in the training set (WEISS, 2010). In addition, it has been observed (TING, 1994; CARVALHO; FREITAS, 2000; ALI; PAZZANI, 1992) that small disjuncts contribute to most classification errors in unseen data. It is likely that small disjuncts are trying to classify small clusters of examples and might be overfitting to them. A study conducted by (JO; JAPKOWICZ, 2004) investigated if the loss in performance in imbalanced data sets was due to the imbalance or a combined effect with small disjuncts. Through a series of experiments with artificial and real-world data sets, the authors concluded that the presence of small clusters of examples in the minority class hinder learning more than the imbalance alone. In addition, they investigated whether it is more effective to address both issues together or individually. They proposed a *data sampling* method called Cluster-Based Over-Sampling (CBO) to address both problems and their preliminary results show that it is worth dealing with both issues together. The most comprehensive investigation into the issue of small disjuncts is the work of (WEISS, 2010). A new measure, named error concentration, is defined by the author and used to assess the impact of small disjuncts on 30 data sets. C4.5 and RIPPER were trained on all data sets and indeed for most situations the majority of errors were concentrated on the small disjuncts. In addition, the author also investigated the relation between small disjuncts and training set size, noise, and data imbalance. In particular, for data imbalance, the author analysed if imbalanced

data sets degrade small disjuncts further. A separate experiment with 26 data sets was conducted and in most cases the error concentration was higher in the original data set than in the balanced data set, which supports the hypothesis that imbalance data sets increases the error rate of small disjuncts.

Another concept of interest when distinguishing between different imbalanced problems is to understand if the imbalance is *intrinsic* or *extrinsic*. An *intrinsic* imbalance means the imbalance is a characteristic of the problem. For instance, in credit card fraud detection most transactions are genuine and only a small percentage is fraudulent (CHAN et al., 1999). Another example is the problem of identifying intrusive traffic in a network, where most traffic is naturally not intrusive (CIESLAK; CHAWLA; STRIEGEL, 2006). On the other hand, an *extrinsic* imbalance means the imbalance observed in a data set is not a characteristic of the problem but rather produced by some external factor. For instance, if data are being streamed from somewhere, interruptions during the stream may introduce an imbalance in the final data received. An *extrinsic* imbalance could also happen due to monetary restrictions during data collection, e.g. few examples could be collected for one of the classes of interest, even though there were plenty of examples to be collected.

2.1.3 Performance Evaluation

Even though Accuracy is the most adopted performance metric to assess a classifier's performance, it is not suitable when the data are imbalanced. For instance, for a two-class problem where the majority class comprises 99% of the examples, a classifier that predicts everything to belong to the majority class achieves an Accuracy of 99%. Such a high Accuracy is misleading and a classifier that did not learn anything about the minority class is hardly useful.

There is not yet a single performance metric to evaluate the performance of classifiers learning from imbalanced data sets. The current best practice among researchers is to analyse a set of performance metrics that capture different aspects of what has been learnt by a classifier (HE; GARCIA, 2009; HAIXIANG et al., 2016).

Similarly to Accuracy, most performance metrics utilised are based on the confusion matrix of a classifier. A confusion matrix, depicted in Figure 2.3, maintains a tally of the correct and incorrect predictions made by a classifier. More specifically, the rows represent the predictions of a classifier while the columns represent the true class of the target variable. A correct prediction about the minority class is called a True Positive (TP) while a wrong prediction about the minority class is called a False Positive (FP). In a similar fashion, True Negative (TN) and False Negative (FP) represent the same idea applied to the majority class.

Accuracy measures the proportion of predictions that were correct and is defined as:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

A classifier's Recall measures the "coverage" of the classifier to the minority class. In

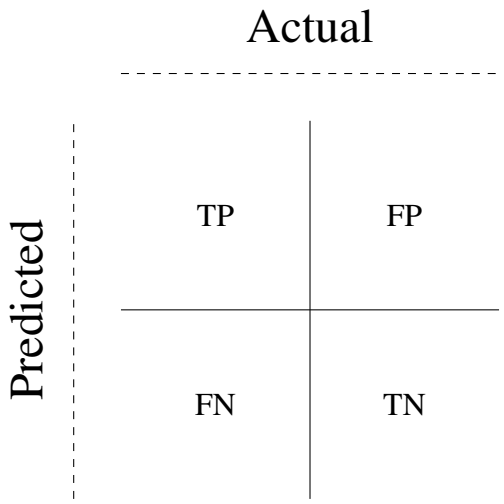

Figure 2.3: A confusion matrix. Examples predicted to belong to the minority class are either correctly classified and labelled as True Positives (TP) or incorrectly classified and labelled as False Positives (FP). The same reasoning applies for the majority class with correctly classified examples labelled as True Negatives (TN) and incorrectly classified examples labelled as False Negatives (FN).

other words, it measures the proportion of examples from the minority class that were correctly classified. Recall is also called *true positive rate*, *sensitivity*, and *hit rate* (FAWCETT, 2006). The same concept applied to the majority class yields the classifier's Specificity.

$$Recall = \frac{TP}{TP+FN}$$

$$Specificity = \frac{TN}{FP+TN}$$

The Precision measures how confident the classifier's predictions are for the minority class. It is the proportion of predictions about the minority class that were correct.

$$Precision = \frac{TP}{TP+FP}$$

The Precision/Recall trade-off can be analysed through the F_1 measure. F_1 is defined as the harmonic mean of Precision and Recall, and is always closer to the smallest value between Precision and Recall. Therefore, a high F_1 can be only achieved if both Precision and Recall are high.

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Many classifiers output a *score* or a *probability* value instead of a hard decision. Varying the decision threshold enables the possibility of drawing ROC curves for these classifiers, which depict the trade-off between Recall and the False Alarm Rate (1 - Specificity). An example of an ROC curve is shown in Figure 2.4. The area under the ROC curve (AUROC) is an important

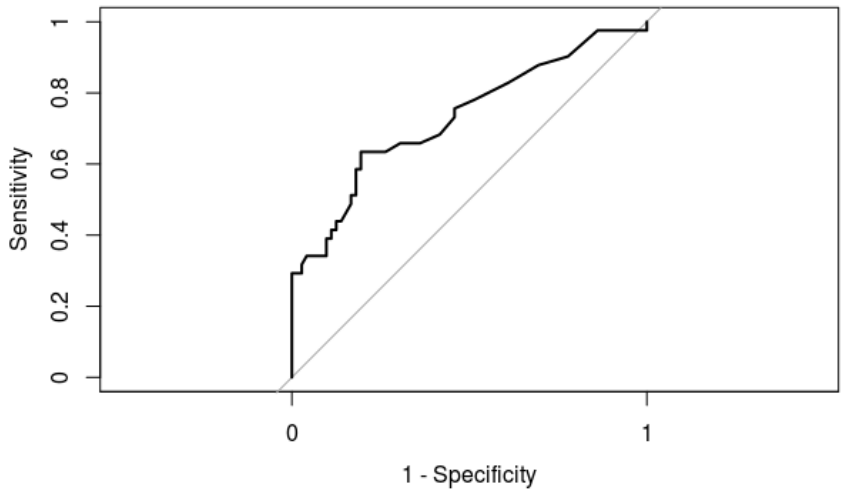

Figure 2.4: An example of a Receiver Operating Characteristic (ROC) Curve.

summary of the ROC curve into a single value. It varies from 0.00 to 1.00 and values closer to 1.00 are more desirable. In addition, the Precision/Recall trade-off can also be analysed by varying the decision threshold, which draws the PR curve, and the AUPR is a convenient way of summarising the PR curve into a single value as well.

If the performance of a classifier on both majority and minority classes is important, the Geometric Mean (G-Mean) is an appropriate measure to evaluate that. It is defined as:

$$\text{G-Mean} = \sqrt{Recall \times Specificity}$$

There are many other performance metrics available (GU; ZHU; CAI, 2009), however we included here only the most employed ones in literature (HAIXIANG et al., 2016).

2.2 Sampling Algorithms

In this section we review the most important *data sampling* algorithms in literature. *Data sampling* techniques can be categorised into over-sampling and under-sampling techniques. Over-sampling techniques increase the size of the minority class by either adding repeated examples or synthesising new examples. On the other hand, under-sampling techniques reduce the size of the majority class by removing examples from the majority class. Hybrid techniques that perform over-sampling and under-sampling are also common.

2.2.1 Under-Sampling

Random Under-Sampling (RUS) is the simplest technique to perform under-sampling. It is a non-heuristic that selects a random sample of examples from the majority class. An example of RUS applied to the imbalanced data set in Figure 2.1 is depicted in Figure 2.5. Another

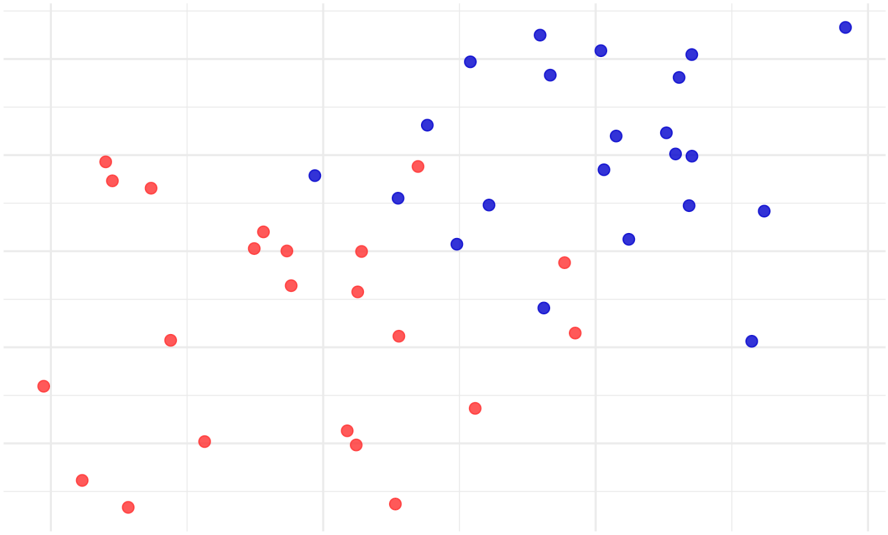

Figure 2.5: The RUS algorithm applied to the imbalanced data set shown in Figure 2.1.

popular method to perform under-sampling is an adaptation of the *k*-Means algorithm. It is called *k*-Means Under-Sampling (KMUS) and works by clustering the majority class with the *k*-Means algorithm and using the computed centroids as the examples to represent the majority class.

One of the first attempts in devising a *data sampling* algorithm to deal with the imbalance in a data set is present in (KUBAT; MATWIN et al., 1997). The authors created an under-sampling technique called One-Sided Selection (OSS), that aims to remove redundant and noisy examples from the majority class. Redundant examples are removed using a variation of the Condensed Nearest Neighbour Rule (CNN) (HART, 2006). The idea of CNN is to select a subset of examples that is consistent with the original data set. A subset of examples is consistent if it correctly classifies all the examples in the original data set using the 1-nearest neighbour rule. To find a consistent subset, OSS starts with a subset containing all examples from minority class and one random example from majority class. Then, it classifies the examples of the original data set using the 1-nearest neighbour rule, and moves to the subset all examples that were misclassified. To remove noisy examples, OSS employs Tomek Links (TLs) (TOMEK, 1976). The idea of a TL is that a pair of examples form a TL if they are nearest neighbours of one another and they belong to different classes. With the consistent subset in hands, OSS then computes all pairs of TLs and, for each pair, removes the example that belongs to the majority class. The behaviour of OSS is depicted in Figure 2.6.

CNN and TLs themselves can also be employed as under-sampling algorithms (BATISTA; PRATI; MONARD, 2004). CNN belongs to a broader category of algorithms called *Prototype Selection* algorithms, and many algorithms in this category are commonly used as under-sampling algorithms. The most used prototype selection technique as an under-sampling technique is the Wilson's Edited Nearest Neighbour Rule (ENN) (WILSON, 1972). The original description of ENN removes all examples that are misclassified by their *k*-nearest neighbours. However, in order to turn it into an under-sampling technique, only examples from the majority class are

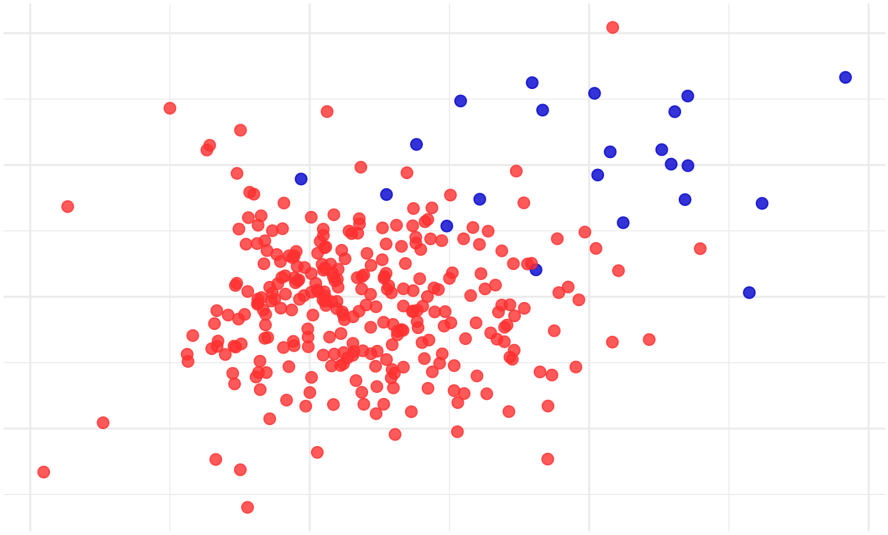

Figure 2.6: The OSS algorithm applied to the imbalanced data set shown in Figure 2.1.

removed. The Neighbourhood Cleaning Rule (NCL) (LAURIKKALA, 2001) is an extension of the ENN idea, where examples of the majority class that contributed to the misclassification of examples of the minority class are removed as well.

A family of under-sampling techniques that take the distance between examples of both classes into account in order to under-sample were proposed in (MANI; ZHANG, 2003). These techniques were named NearMiss-1 (NM1), NearMiss-2 (NM2), NearMiss-3 (NM3), and Most Distant (MD). NM1 selects the majority examples that are on average closer to their k-nearest neighbours relative to the minority class, while MD selects the majority examples that are further. On the other hand, NM2 selects the majority examples that are on average closer to their k-furthest neighbours relative to the minority class. Unlike the others, NM3 selects for each minority class example a number of its k-nearest neighbours relative to the majority class.

Algorithms based on clustering are common as well. In the work of (YEN; LEE, 2009) Under-Sampling Based on Clustering (SBC) was proposed. SBC clusters both majority and minority classes together and, for each cluster, selects a random sample of examples from the majority class based on the imbalance ratio of the cluster. In addition, the authors proposed five variations of SBC, where each of them selects the examples from the majority class in a non-random way. Four of them are based on the NearMiss family of algorithms (MANI; ZHANG, 2003) and are named SBCNM-1, SBCNM-2, SBCNM-3, and SBCMD. The fifth variation is called Under-Sampling Based on Clustering with Most Far (SBCMF) and selects the majority class examples whose average distance to all minority class examples are the biggest. An illustration of SBC is shown in Figure 2.7. Instead of clustering both classes together as in SBC, Diversified Sensitivity Based Under-Sampling (DSUS) (NG et al., 2015) clusters both classes separately and selects examples from both classes in an iterative manner, using a sensitivity measure and a RBFNN. To account for highly non-linear data sets, Density-Based Majority Under-Sampling Technique (DBMUTE) (BUNKHUMPORNPAT; SINAPIROMSARAN, 2017)

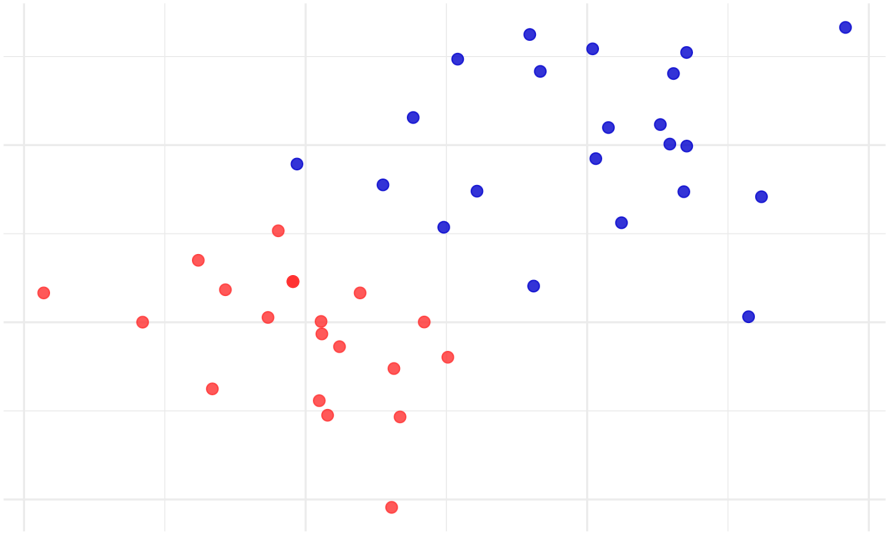

Figure 2.7: The SBC algorithm applied to the imbalanced data set shown in Figure 2.1.

uses the DBSCAN clustering algorithm to remove majority examples in the overlapping region between the classes. The authors define the concept of a "blemished" example and use this concept to remove examples from the majority class. A blemished example is a majority example that is near one of the centroids of the minority class, according to DBSCAN.

2.2.2 Over-Sampling

Random Over-Sampling (ROS) is analogous to RUS for over-sampling. It selects examples from the minority class at random and append them to the data.

The imbalance in a data set may also happen *within* a class, not just *between* the classes. The Cluster-Based Over-Sampling (CBO) technique proposed by (JO; JAPKOWICZ, 2004) aims to deal with both imbalances. It works by first clustering the classes using the k-Means algorithm, then increasing the size of the clusters in the majority class until they have the same size as the biggest cluster in the majority class, and finally increasing the size of the clusters in the minority class until they all have the same size and their combined size is the same as the majority class. Individual clusters have their size increased by the ROS technique.

ROS is a simple technique to increase the size of the minority class, however it does not add any new information to the minority class. The first over-sampling technique to add new examples to the minority class was the Synthetic Minority Over-Sampling TEchnique (SMOTE) developed by (CHAWLA et al., 2002). SMOTE works by synthesising new examples in the line segment connecting two nearest neighbours in the minority class. The experiments in (CHAWLA et al., 2002) showed that the SMOTE technique succeeded in increasing the recognition of the minority class, which led to many adaptations of the technique. SMOTE is illustrated in Figure 2.8

The first adaptation of SMOTE was the Borderline-SMOTE (BDL-SMOTE) algorithm

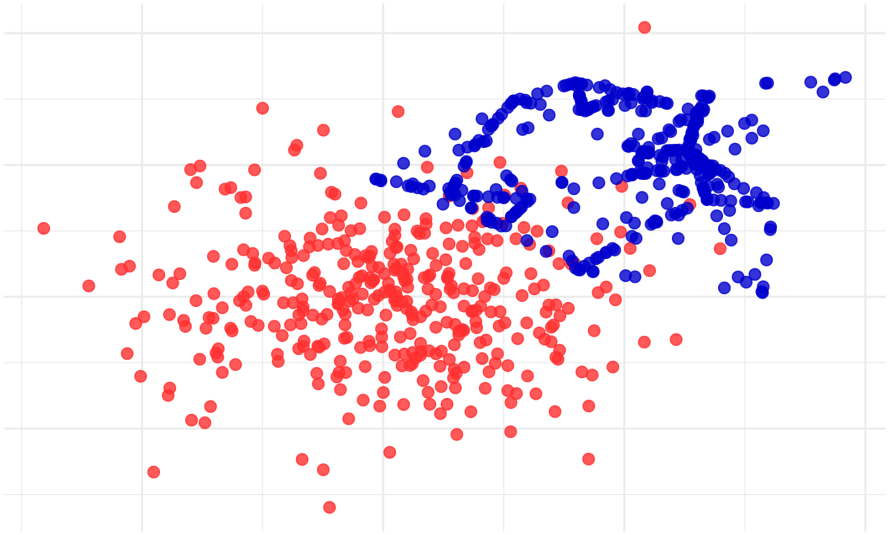

Figure 2.8: The SMOTE algorithm applied to the imbalanced data set shown in Figure 2.1.

(HAN; WANG; MAO, 2005). BDL-SMOTE works similarly to SMOTE, however it only synthesises new examples near the border of the classes. The authors believe that examples near the border are more likely to be misclassified, hence the need to generate more examples of the minority class near the border. In a similar spirit, the Adaptive Synthetic Sampling (ADASYN) technique was proposed by (HE et al., 2008). While BDL-SMOTE synthesises new examples near the border, ADASYN generates new examples near the examples that are considered hard to learn. The learning hardness is measured according to the number of neighbours that belong to the majority class, where more neighbours belonging to the majority class means a harder to learn minority example. The only exception happens when all neighbours belong to the majority class, in that case the minority example is considered noise and no new examples are synthesised near this example. Applying different levels of sampling to different minority examples is an idea present in the BDL-SMOTE and the ADASYN techniques, and it is exploited further in the Safe-Level-SMOTE (SL-SMOTE) (BUNKHUMPORNPAT; SINAPIROMSARAN; LURSINSAP, 2009). SL-SMOTE associates a level of safeness to each minority example, and based on these safe levels synthesises new examples in different manners. The level of safeness of each example is computed as the number of examples that belong to the minority class among the k-nearest neighbours. Density-Based Synthetic Minority Over-sampling TEchnique (DBSMOTE) was proposed in (BUNKHUMPORNPAT; SINAPIROMSARAN; LURSINSAP, 2012) and uses the DBSCAN clustering algorithm to aid in the synthesis of new examples. It works by clustering the minority class with DBSCAN, constructing a directly density-reachable graph and computing the pseudo-centroid of each cluster, and finally synthesising new examples along random edges of the shortest path between minority examples and pseudo-centroids.

Despite all the effort that SMOTE and related adaptations put into generating new examples for the minority class, noisy examples either already present or introduced by the algorithms could still impact the learning process. In (BATISTA; PRATI; MONARD, 2004), two

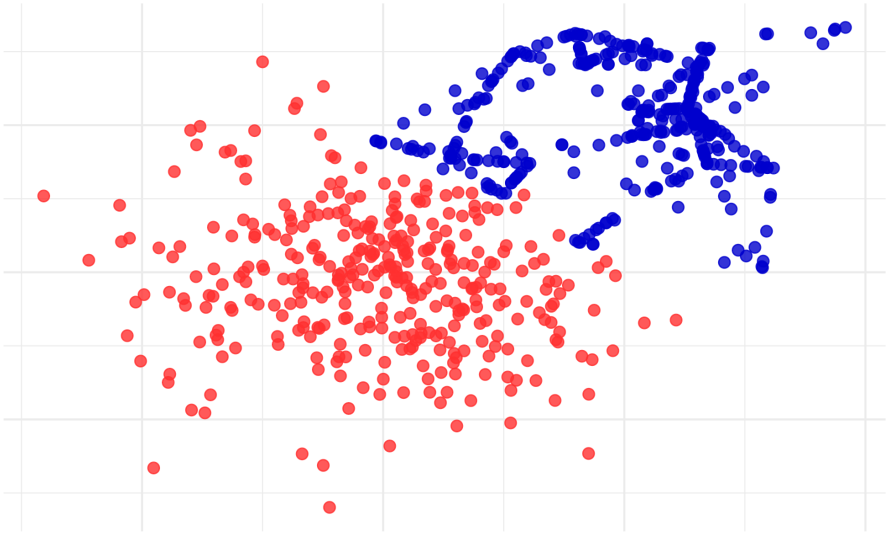

Figure 2.9: The SMOTE + ENN algorithm applied to the imbalanced data set shown in Figure 2.1.

versions of SMOTE with an additional cleaning step were proposed. The cleaning algorithms, ENN and TL, were used to remove examples of both classes after over-sampling, and the two proposed algorithms were named SMOTE + ENN and SMOTE + TL. An example of the behaviour of SMOTE + ENN is shown in Figure 2.9. In a similar fashion, (SÁEZ et al., 2015) proposed the use an ensemble-based filter to remove noisy examples after over-sampling with SMOTE. More specifically, the authors employed the Iterative-Partitioning Filter (IPF) (KHOSHGOFTAAR; REBOURS, 2007) and named the proposed algorithm SMOTE + IPF. Instead of removing noisy examples after over-sampling, (RIVERA, 2017) proposes Noise Reduction A Priori Synthetic Over-Sampling (NRAS), a new algorithm that removes noisy examples from the minority class prior to over-sampling. NRAS works by removing examples from the minority class that have a probability of belonging to the minority class smaller than a threshold. The filtered data set is then over-sampled using SMOTE.

Although SMOTE and related adaptations manage to synthesise new examples for the minority class, these new examples might not belong to the underlying density distribution from where the original examples come from. In (GAO et al., 2014) the authors propose Probability Density Function Estimation Based Over-Sampling (PDFOS), an algorithm that synthesises new examples based on the estimated density function of the minority class. The density function of the minority class is computed using a Parzen Window (PW) method (PARZEN, 1962). Based on the aforementioned observations and the fact that SMOTE does not sufficiently expand the decision boundary, (ZHANG; LI, 2014) propose the Random Walk Over-Sampling (RWO). The technique is based on the Central Limit Theorem and generates new examples by randomly perturbing the minority examples. Following the same idea of generating new examples that come from the underlying density distribution, (DAS; KRISHNAN; COOK, 2015) propose the RApidly COnverging Gibbs Sampler (RACOG) and wRACOG techniques. RACOG utilises

Gibbs sampling to generate new examples from the estimated joint density distribution of the minority class. The second technique, wRACOG, is a wrapper around RACOG that gets rid of RACOG hyperparameters. Unfortunately, RACOG and wRACOG only work for discrete attributes.

Following a different approach, (BELLINGER, 2016) proposes a framework for the generation of new examples for data sets that conform to the "manifold property". The framework first induces the manifold structure based on the data set, then over-samples new examples in the embedded space, and reverse-map the newly synthesised examples back to feature space. To induce the structure of the manifold the authors tested the Principal Component Analysis (PCA) and Autoencoder (AE) algorithms, however any manifold learning algorithm could be employed.

2.3 Summary

In this chapter we reviewed the problem of learning from imbalanced data sets and the most important *data sampling* algorithms.

We started by defining what is an imbalanced data set and highlighted that the imbalance is not the only source of difficulty when learning from imbalanced data sets. Other sources of complexity such as noise, overlap between classes, multimodal density distributions, and small sample size, are usually present in imbalanced data sets and degrade performance of classifiers learning from imbalanced data sets further. Then, in addition to the prevalent *between-class* imbalance we also discussed the *within-class* imbalance, caused by underrepresented modes of multimodal density distributions. Since the *within-class* imbalance is rarely discussed under the light of multimodal density distributions, we briefly reviewed the closely related issue of small disjuncts in imbalanced data sets. In addition to the two types of imbalance, we commented on the nature of the imbalance, which could be *intrinsic* or *extrinsic* to the problem. Finally, we showed how inappropriate the Accuracy performance metric is to evaluate classifiers learning from imbalanced data sets and analysed a number of alternative performance metrics that assess the performance of a classifier from many different points of view.

After reviewing the main issues associated with classifiers learning from imbalanced data sets, we discussed the most important under-sampling and over-sampling algorithms. Under-sampling algorithms present many strategies to reduce the imbalance in the data by removing examples from the majority class, either in a non-heuristic or heuristic way. RUS is the simplest under-sampling algorithm and removes examples from majority class at random. Other approaches try to incorporate information present in the data to select the examples from majority class in a non-random way. For instance, the NearMiss family of algorithms use the distance between examples of the two classes to select the examples from majority class, while SBC, DSUS, and DBMUTE cluster the data and employ different criteria to select the examples from majority class. On the other hand, over-sampling algorithms increase the size of the minority class by adding examples to it. The simplest over-sampling algorithm is ROS, which randomly

selects examples from the minority class and appends them to the data. The main drawback of ROS is the fact that it does not add new examples to the data, an issue that was first remedied by the SMOTE algorithm. SMOTE randomly synthesises new examples along the line segment connecting nearest neighbours of the minority class. The success and drawbacks of SMOTE led to many adaptations of it. BDL-SMOTE, ADASYN, SL-SMOTE, and DBSMOTE generate new examples like SMOTE, along the line segment joining nearest neighbours of the minority class, however they do it following different criteria. In addition to adaptations of SMOTE, several methods added cleaning steps to SMOTE, either by removing noise before over-sampling as in NRAS or by removing noise after over-sampling as in SMOTE + ENN. Since SMOTE based algorithms potentially generate examples that do not belong to the underlying density distribution of the minority class, several algorithms were devised with this issue in mind. For instance, PDFOS estimates the probability density function of the minority class and synthesises new examples according to it, while RWO is based on the Central Limit Theorem and synthesises new examples by perturbing examples from the minority class.

3

EXPERIMENTAL METHODOLOGY

In this chapter we report the general experimentation settings that were utilised to assess the usefulness of RRUS and *k*-INOS. These include the imbalanced data sets, the base classifiers, and the performance metrics employed. As the proposed algorithms are fundamentally different, details about the comparison of their performance to other sampling algorithms are postponed to the appropriate chapters.

3.1 Training Process

A sampling algorithm modifies the input training set by adding examples to it or removing examples from it. Since there is no way of comparing the quality of the old and the new training sets, we cannot evaluate the performance of a sampling algorithm by solely comparing the input and output training sets. Therefore, to analyse the performance gain of a sampling algorithm, a classifier must be trained on the old and the new training sets and the performance difference of the classifier on a test set should be used as a proxy to the performance gain of the sampling algorithm.

To fully evaluate a sampling algorithm, three components are necessary: an imbalanced data set, a classifier, and a performance metric. However, due to chance, a bad sampling algorithm may just work well for a specific combination of imbalanced data set, classifier, and performance metric, which may lead to wrong claims about the sampling algorithm. Therefore, to avoid drawing unreasonable conclusions about the proposed algorithms, a selection of 50 imbalanced data sets, 6 classifiers, and 7 performance metrics were utilised during the experiments. In addition, a 5×2-fold cross validation was used to compute mean values for the performance metrics for all combinations of sampling algorithm, imbalanced data set, and classifier.

3.2 Data Sets

There is usually little overlap in the data sets employed in the test of *data sampling* algorithms found in the literature (HE et al., 2008; BUNKHUMPORNPAT; SINAPIROMSARAN; LURSINSAP, 2009; ZHANG; LI, 2014). Moreover, in some cases, conclusions about proposed

algorithms are drawn from experimentation with a small number of data sets. To overcome this, 50 data sets were employed here with the constraints of having, after cleaning, at least 40 examples in the minority class, no missing values, a maximum of 60 features (all numerical), and a minimum imbalance ratio (IR) of 1.50. IR is defined as the ratio of the number of examples in the majority class to the number of examples in the minority class.

The 50 data sets come from several sources and are a superset of the data sets utilised in many related works. To avoid optimistic measurements of the performance metrics, all data sets were cleaned up by eliminating constant features, repeated examples, and inconsistent examples (in this sequence). The only exceptions were the data sets from the Promise Repository (MENZIES; KRISHNA; PRYOR, 2015), which were already cleaned up by (SHEPPERD et al., 2013). Two examples are considered inconsistent if they belong to different classes but have the same values for their features. Table 3.1 contains detailed information about the 50 data sets chosen.

3.3 Classifiers

The classifiers employed were Logistic Regression (LR) (ABU-MOSTAFA; MAGDON-ISMAIL; LIN, 2012), *k*-Nearest Neighbours (*k*-NN) (COVER; HART, 1967), C4.5 Decision Tree (DT) (QUINLAN, 2014), Support Vector Machine (SVM) with Gaussian Kernel (SCHÖLKOPF; SMOLA, 2002), Random Forest (RF) (BREIMAN, 2001), and Gradient Boosting Machine (GBM) (FRIEDMAN, 2001).

This selection of classifiers was made to ensure diversity, as we have a linear classifier (LR), an instance-based classifier (*k*-NN), a tree-based classifier (DT), a kernel-based classifier (SVM), a bagging-based ensemble (RF), and a boosting-based ensemble (GBM). We list below the specific implementations utilised.

- LR: R interface to *LIBLINEAR* (HELLEPUTTE, 2017).
- *k*-NN: R's *kknn* package (SCHLIEP; HECHENBICHLER, 2016)
- DT: R interface to *Weka's* J48 (HORNIK; BUCHTA; ZEILEIS, 2009).
- SVM: R's *kernlab* package (KARATZOGLOU et al., 2004).
- RF: R's *randomForest* package (LIAW; WIENER, 2002).
- GBM: R interface to the *eXtreme Gradient Boosting* system (CHEN et al., 2017).

For most classifiers the default hyperparameters provided by the associated implementations were used with a few exceptions. These exceptions were either because there was no default value or the default value was not a commonly used one. In particular *k*-NN had *k* set to 3 and *distance* set to Euclidean, and GBM had the number of *boosting iterations* set to 50.

Table 3.1: Data sets employed with corresponding references. All data sets were cleaned up by removing constant features, repeated examples, and inconsistent examples (in this order). Numbers shown are after data cleansing. Entries are sorted by increasing values of IR.

Name	# Examples	# Features	IR
Spambase (LICHMAN, 2013)	4204	57	1.5084
Ionosphere (LICHMAN, 2013)	350	33	1.8000
glass1 (ALCALÁ et al., 2010)	213	9	1.8026
fourclass (CIESLAK et al., 2012)	862	2	1.8078
MC2" (MENZIES; KRISHNA; PRYOR, 2015)	125	39	1.8409
ecoli-0_vs_1 (ALCALÁ et al., 2010)	220	6	1.8571
Pima Indians Diabetes(LICHMAN, 2013)	768	8	1.8657
MAGIC Gamma Telescope (LICHMAN, 2013)	18905	10	1.8762
SVMguide1 (CIESLAK et al., 2012)	3025	4	1.9512
QSAR biodegradation (LICHMAN, 2013)	1052	41	1.9718
glass0 (ALCALÁ et al., 2010)	213	9	2.0870
Vertebral Column (2C) (LICHMAN, 2013)	310	6	2.1000
yeast1 (ALCALÁ et al., 2010)	1453	8	2.4188
phoneme (CIESLAK et al., 2012)	5395	5	2.4211
PC5" (MENZIES; KRISHNA; PRYOR, 2015)	1711	38	2.6327
Haberman's Survival (LICHMAN, 2013)	277	3	2.7945
vehicle2 (ALCALÁ et al., 2010)	846	18	2.8807
Blood Transfusion Service Center (LICHMAN, 2013)	471	4	2.9915
Parkinsons (LICHMAN, 2013)	195	22	3.0625
glass-0-1-2-3_vs_4-5-6 (ALCALÁ et al., 2010)	213	9	3.1765
vehicle0 (ALCALÁ et al., 2010)	846	18	3.2513
ecoli1 (ALCALÁ et al., 2010)	336	7	3.3636
JM1" (MENZIES; KRISHNA; PRYOR, 2015)	7782	21	3.6543
ecoli2 (ALCALÁ et al., 2010)	336	7	5.4615
segment0 (ALCALÁ et al., 2010)	2084	18	6.0405
PC4" (MENZIES; KRISHNA; PRYOR, 2015)	1287	37	6.2712
PC3" (MENZIES; KRISHNA; PRYOR, 2015)	1077	37	7.0373
estate (CIESLAK et al., 2012)	5211	12	7.4048
yeast3 (ALCALÁ et al., 2010)	1453	8	7.9691
yeast-2_vs_4 (ALCALÁ et al., 2010)	489	8	8.5882
pendigits (CIESLAK et al., 2012)	10992	16	8.6252
yeast-0-2-5-6_vs_3-7-8-9 (ALCALÁ et al., 2010)	977	8	8.9694
yeast-0-2-5-7-9_vs_3-6-8 (ALCALÁ et al., 2010)	977	8	8.9694
yeast-0-3-5-9_vs_7-8 (ALCALÁ et al., 2010)	504	8	9.0800
satimage (CIESLAK et al., 2012)	6430	36	9.2880
yeast-0-5-6-7-9_vs_4 (ALCALÁ et al., 2010)	527	8	9.3333
page-blocks0 (ALCALÁ et al., 2010)	5383	10	9.6806
vowel0 (ALCALÁ et al., 2010)	988	13	9.9778
PC1" (MENZIES; KRISHNA; PRYOR, 2015)	705	37	10.5574
covtype (CIESLAK et al., 2012)	38500	10	13.0153
oil (CIESLAK et al., 2012)	937	48	21.8537
letter (CIESLAK et al., 2012)	18668	16	23.6931
winequality-red-4 (ALCALÁ et al., 2010)	1359	11	24.6415
compustat (CIESLAK et al., 2012)	13655	20	25.2596
yeast4 (ALCALÁ et al., 2010)	1453	8	27.4902
ism (CIESLAK et al., 2012)	7845	6	30.0079
mammography (WOODS et al., 1993)	7847	6	30.0158
yeast5 (ALCALÁ et al., 2010)	1453	8	32.7907
MC1" (MENZIES; KRISHNA; PRYOR, 2015)	1988	38	42.2174
shuttle-2_vs_5 (ALCALÁ et al., 2010)	3316	9	66.6735

3.4 Performance Metrics

As discussed before in Chapter 2, Accuracy alone is not suitable to evaluate the performance of classifiers learning from imbalanced data sets. And, as we want to evaluate two different types of *data sampling* algorithms, it is important to inspect their performance from many different perspectives.

Therefore, in order to provide a thorough analysis of the behaviour of the proposed algorithms, we selected 7 performance metrics. The selected performance metrics were: Accuracy, AUROC, F_1, Precision, Recall, G-Mean, and Specificity. These metrics are among the most used ones in literature (HAIXIANG et al., 2016) and are explained in detail in Subsection 2.1.3.

4

REPEATED RANDOM UNDER-SAMPLING

In this chapter we propose RRUS, short for *Repeated Random Under-Sampling*, an under-sampling algorithm more likely to select a subset of examples from the majority class with an estimated density distribution closer to the density distribution of the majority class. RRUS accounts for multimodal density distributions and is loosely based on the KMUS and RUS algorithms. The main idea of RRUS is to break the majority class into several clusters, sample many random subsets of examples from each cluster, and select the subsets that best preserve the density distribution of the majority class according to a heuristic. The experiments compared RRUS to other three under-sampling algorithms and the results showed that RRUS outperformed them, frequently in a statistically significant manner, for most classifiers and performance metrics evaluated.

The rest of this chapter is organised as follows. Section 4.1 reviews the works that have a proposal similar to RRUS and puts it into perspective. Section 4.2 explains the proposed algorithm in great detail through several examples and a concise algorithmic description of it. Section 4.3 highlights the experimental settings specific to the evaluation of RRUS, and section 4.4 presents the results of the experimentation and comments on them. Section 4.5 compares the performance gains attained by RRUS on all data sets against the RUS and KMUS algorithms. Finally, Section 4.6 summarises the main findings and draws the conclusions.

4.1 Related Work

In this section we review the main articles that have proposals similar to RRUS. These include under-sampling algorithms that are designed to preserve the density distribution of the majority class or are based on clustering techniques. Note that all articles mentioned in this section have been reviewed in Chapter 2. Nevertheless, we describe the main ideas again and include information about the experiments conducted by the authors.

In (YEN; LEE, 2009) the authors proposed Under-Sampling Based on Clustering (SBC). SBC clusters both majority and minority classes together, and for each cluster, selects a random sample of examples from the majority class based on the imbalance ratio of the cluster. Moreover, the authors proposed 5 variations of SBC that select the examples of the majority class in a

non-random way. SBC and its variations were compared to other 2 under-sampling algorithms and the no sampling strategy. The experimentation included 2 real-world data sets, which were split into 80% for training and 20% for evaluation, a *Neural Network* (NN) classifier, and 3 performance metrics. Compared to the other algorithms, SBC achieved higher values of Precision and F_1.

In (NG et al., 2015) the authors proposed Diversified Sensitivity Based Under-Sampling (DSUS). DSUS clusters both majority and minority classes separately and repeatedly selects examples from both classes using a stochastic Sensitivity Measure (SM) (YEUNG et al., 2007) coupled with a RBFNN. The experimentation conducted to evaluate the usefulness of DSUS comprised 14 data sets, which were randomly split 30 times into 50% for training and 50% for evaluation, 2 classifiers, and 3 performance metrics. DSUS was compared to other 6 sampling algorithms and the results showed that it performed better than the alternatives most of the time, especially in regard to the F_1 metric.

In (BUNKHUMPORNPAT; SINAPIROMSARAN, 2017) the authors proposed Density-Based Majority Under-Sampling Technique (DBMUTE). DBMUTE clusters the minority class using DBSCAN and uses the idea of "blemished" examples to remove unwanted examples from the majority class. The experimentation compared DBMUTE to other 9 sampling algorithms and the no sampling strategy, and the algorithms were tested on 9 data sets, which were randomly split 3 times into 2/3 for training and 1/3 for evaluation. 8 base classifiers were employed and their performance evaluated on 2 performance metrics. The results showed that in many cases DBMUTE had the best results for both AUROC and F_1.

A noticeable trend in these recent works is the clustering of the data followed by some criterion to select examples from the majority class. However, the selection step is usually random and therefore highly variable. RRUS address this issue by repeatedly performing the random step and using a heuristic to select the best subset of examples.

4.2 RRUS

Under-sampling algorithms by definition discard examples from the majority class in order to reduce the imbalance present in the data. For any desired level of imbalance between the majority and minority classes after under-sampling, if m^- is the number of examples in the majority class and $m < m^-$ is the number of examples to be selected from the majority class, there are $\binom{m^-}{m}$ possible subsets of examples. Even for relatively small data sets with a mild imbalance, the number of subsets can be enormous. For instance, if $m^- = 50$ and $m = 20$, then $\binom{50}{20} > 10^{13}$.

Example 1. Given a set of examples drawn from a known distribution, different subsets of examples may not reflect the original distribution. Consider a set of 400 examples where 100 of them are drawn from a Gaussian distribution with $\mu = -2$ and $\sigma = 1$ and the other 300

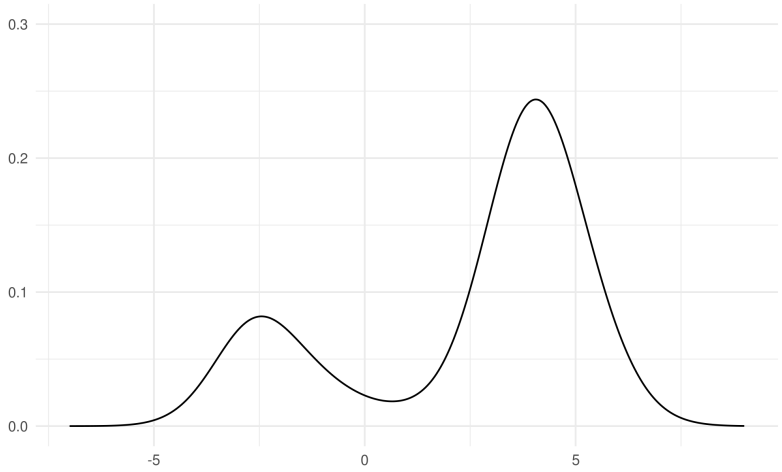

Figure 4.1: Estimated density distribution of 400 examples where 300 were drawn from a Gaussian distribution with $\mu = 4$ and $\sigma = 1$ and the other 100 were drawn from a Gaussian distribution with $\mu = -2$ and $\sigma = 1$.

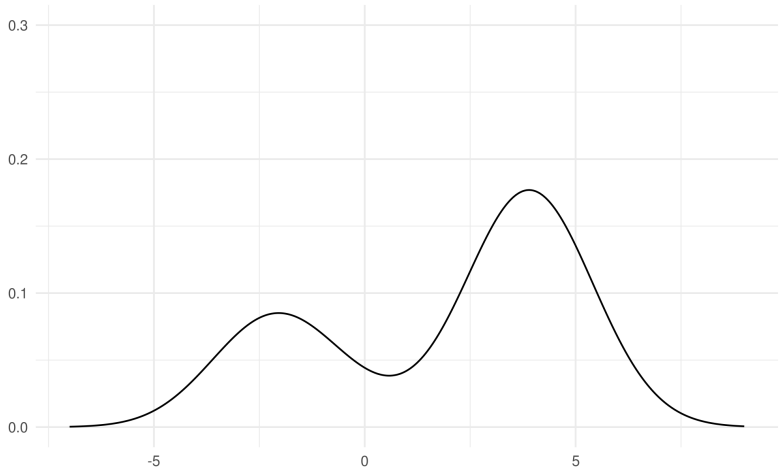

Figure 4.2: Estimated density distribution of a random subset of 30 examples from the data in Figure 4.1

examples are drawn from a Gaussian distribution with $\mu = 4$ and $\sigma = 1$. Figure 4.1 plots the estimated density distributions of the 400 examples and, Figures 4.2 and 4.3 plot the estimated density distributions of two random subsets of 30 examples. As it can be seen, the subsets do not accurately reflect the original distribution, since they were selected at random.

Selecting a subset of examples from the majority class that accurately reflects the density distribution of the majority class is important in order to minimise the amount of knowledge lost about the majority class. The benefits are two-fold: a classifier trained on such a reduced data set has an increased chance of learning an appropriate representation for the minority class and still maintain a high accuracy on the majority class.

RRUS is an enhancement to the RUS algorithm that increases the likelihood of selecting a subset of examples from the majority class that more faithfully reflects the majority class. RRUS selects many random subsets of examples from the majority class and uses a heuristic

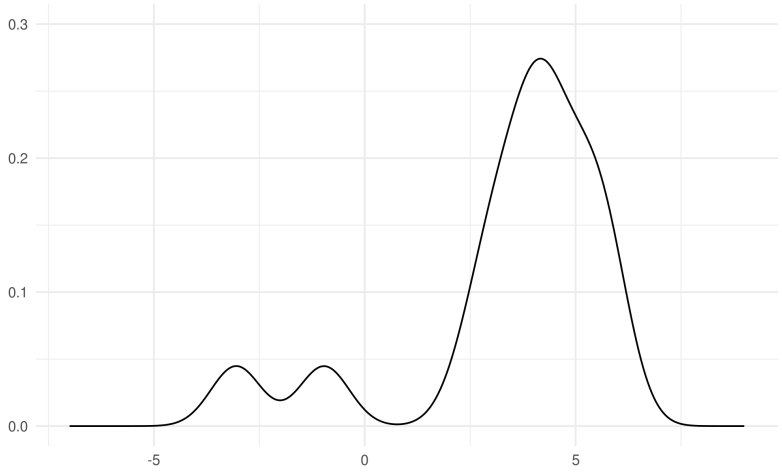

Figure 4.3: Estimated density distribution of a random subset of 30 examples from the data in Figure 4.1

to choose the subset that has an estimated density distribution mostly similar to the majority class. As the density distribution of the majority class can be multimodal, the majority class is first clustered using the k-Means algorithm and the aforementioned procedure is applied to each cluster.

The RRUS algorithm has two main steps: cluster the majority class and select a subset of examples from each cluster. First, the majority class is clustered using the k-Means algorithm with k being a hyperparameter of RRUS. This step is performed in order to account for multimodal distributions in the majority class. Next, assume that m^- is the number of examples in the majority class, m_i^- is the number of examples in the i-th cluster, $m < m^-$ is the number of examples to select from the majority class, and $m_i < m_i^-$ is the number of examples to select from the i-th cluster. Then, each m_i is computed as $m_i = \left\lceil m \times \frac{m_i^-}{m^-} \right\rceil$, where m is a hyperparameter of RRUS and determines the level of imbalance after under-sampling. Finally, $times$ random subsets are sampled from each cluster, and the subset with the centroid closest to the centroid of the respective cluster is chosen. The Euclidean distance is considered to compute the distance between the centroids. A concise algorithmic description of RRUS is given in Algorithm 1.

Example 2. Using the same set of examples from Example 1, Figure 4.4 shows the estimated density distribution of 30 examples selected using the RRUS algorithm. This time, the estimated density distribution is much closer to the density distribution in Figure 4.1.

RRUS has three hyperparameters: m, k, and $times$. m is a common hyperparameter among under-sampling algorithms and determines how many examples are selected from the majority class. Usually m is set to m^+, the number of examples in the minority class, therefore balancing the number of examples in both classes. k controls the number of clusters to be computed by the k-Means algorithm. Ideally, k should be set to the number of modes in the underlying density distribution of the majority class, however this information is rarely known and the best value for k must be investigated. $times$ controls how many random subsets are sampled from each cluster

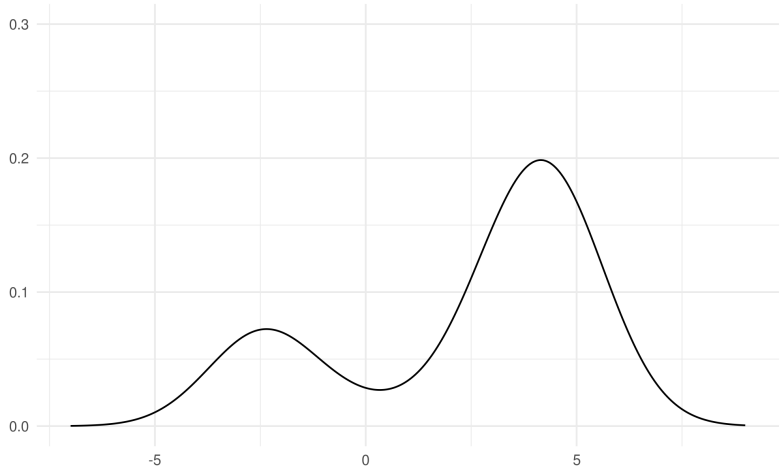

Figure 4.4: Estimated density distribution of 30 examples selected from the data described in 4.1 using RRUS. The values for RRUS hyperparameters were $k = 2$ and $times = 100$.

and the higher its value the more likely RRUS is to select an appropriate subset of examples. However, the higher the value of *times* the longer it takes RRUS to finish execution.

Algorithm 1: RRUS

Objective

Select a subset of examples from the majority class that preserves the density distribution of the majority class.

Input

- D: An imbalanced data set.
- m: Number of examples to select from the majority class.
- k: Number of clusters to compute in the majority class with the k-Means algorithm.
- $times$: Number of random subsets to sample from each cluster.

Output

- $D*$: A more balanced version of D.

Algorithm

1. Cluster the majority class using the k-Means algorithm.
2. For each cluster, select $times$ random subsets of examples from it. The size of the subsets from the i-th cluster is $m_i = \left\lceil m \times \frac{m_i^-}{m^-} \right\rceil$.
3. Select the subsets that have a centroid closer to the respective cluster centroid.

4.3 Experimental Methodology

To evaluate the benefits of using RRUS we compared it to other 3 similar under-sampling algorithms. The experimental framework utilised, which includes 50 imbalanced data sets, 6 classifiers, and 7 performance metrics, is described in Chapter 3. For each combination of classifier and performance metric, the *Friedman Aligned Ranks* test (HODGES; LEHMANN et al., 1962) followed by the *Finner* post hoc test (with p-value adjustment) (FINNER, 1993) were used to judge the statistical significance of the results achieved by RRUS. A significance level of 5% was used and RRUS was the control algorithm in the *Finner* post hoc test. These statistical tests were selected following the guidelines provided in the work of (GARCÍA et al., 2010).

The under-sampling algorithms chosen to be compared to RRUS were RUS, KMUS, and SBC. All algorithms were set to balance the distribution of examples. The algorithms that depend on k-Means, namely KMUS, SBC, and RRUS, had k-Means' maximum number of *iterations* set to 100 and number of *random restarts* set to 10, which are common values in practice. After some experimentation with RRUS, we set its k value to 9 and $times$ to 100. Since SBC's authors did not suggest a value for SBC's k, it was set to 9 as well for a fair comparison. Note that these values were the same for all data sets to make a fair assessment.

4.4 Results and Discussion

The mean ranks obtained by the *Friedman Aligned Ranks* test are shown in Table 4.1. The results are broken down by classifier and, for each performance metric, the mean ranks can vary from 1.00 to 4.00. A mean rank closer to 1.00 means the corresponding under-sampling algorithm had the highest performance most of the time, while a mean rank closer to 4.00 means the opposite. For each performance metric, a value in bold highlights the best ranked algorithm, an underlined value means that RRUS' performance was significantly better than the underlined algorithm, and a crossed out value means that RRUS' performance was significantly worse than at least one other under-sampling algorithm.

Overall, RRUS consistently obtained the best mean rank for most classifiers and performance metrics with the exceptions of 3-NN classifier and Recall metric. In addition, RRUS' performance was frequently significantly better than KMUS and SBC, and many times significantly better than RUS. For every metric and classifier, apart from Recall, RRUS always ranked better than RUS and SBC.

For the LR and DT classifiers RRUS achieved the best mean rank for all performance metrics except for Recall. RRUS was significantly better than KMUS and SBC for Accuracy and Specificity for both classifiers. In addition, for the F_1, Precision, and G-Mean metrics, RRUS was significantly better than KMUS and SBC for the DT classifier, and significantly better than SBC for the LR classifier. While RRUS significantly outperformed all other under-sampling

algorithms for the AUROC metric and the LR classifier, it only significantly outperformed KMUS and SBC for the DT classifier. For the Recall metric, RRUS was significantly worse than SBC for the LR classifier, and significantly worse than SBC and KMUS for the DT classifier.

Regarding the SVM classifier, RRUS ranked best for the AUROC, F_1, and G-Mean metrics, and significantly outperformed all other under-sampling algorithms in these metrics. Even though RRUS did not achieve the best mean rank for the Accuracy, Precision, Recall, and Specificity metrics, it was still significantly better than RUS and SBC for the Accuracy, Precision, and Specificity metrics, and significantly better than KMUS for the Recall metric.

The RF and GBM classifiers had similar results for all performance metrics. RRUS ranked best for all performance metrics, except for Recall, and it was significantly better than KMUS and SBC in these ones. For both classifiers, RRUS was significantly worse than SBC and KMUS for the Recall metric. The similar performance is likely due to the fact that both classifiers are tree-based ensembles, and the ensemble type (bagging or boosting) did not influence considerably.

For 3-NN, unlike the other classifiers, RRUS did not achieve the best mean rank for most performance metrics. Nevertheless, RRUS had the second best mean rank for Accuracy, AUROC, F_1, Precision, and Specificity, and significantly outperformed RUS and SBC in all of them. Moreover, RRUS ranked best for the G-Mean metric and it was significantly better than SBC. RRUS was significantly worse than KMUS for the AUROC and Specificity metrics, and significantly worse than SBC for the Recall metric.

A summary of the results in Table 4.1 are shown in Tables 4.2 and 4.3. Both tables count the number of times RRUS had the 1st, 2nd, 3rd, or 4th best mean rank either from the point of view of the classifiers or the point of view of the performance metrics. Table 4.2 represents the point of view of the classifiers. For the LR, DT, RF, and GBM classifiers RRUS achieved the best mean rank 6 times and the worst mean rank 1 time. For the SVM classifier RRUS had a balanced performance, achieving the best mean rank 3 times and the second best mean rank 4 times, while for the 3-NN classifier it achieved the second best mean rank 5 times and the first and third best mean ranks 1 time each. For none of the classifiers RRUS achieved the best mean rank for all performance metrics. Table 4.3 represents the point of view of the performance metrics. For the G-Mean metric, RRUS achieved the best mean rank for all classifiers, while for the AUROC and F_1 metrics RRUS achieved the best mean rank 5 times and the second best mean rank 1 time. For the Accuracy, Precision, and Specificity metrics RRUS achieved the best mean rank 4 times and the second best mean rank 2 times, while for the Recall metric RRUS mostly attained the worst mean rank.

4.5 RRUS versus RUS, KMUS, and SBC

As mentioned before, RRUS is loosely based on the RUS and KMUS algorithms. Thus, it is natural to wonder how much of an improvement RRUS attains over them. To answer this

Table 4.1: Mean ranks of the *Friedman Aligned Ranks* test. The results are split by classifier and, for each performance metric, the values can vary from 1.00 to 4.00. Values in bold indicate the best ranked algorithm, underlined values highlight the under-sampling algorithms that RRUS' performance was significantly better, and crossed out values indicate that RRUS' performance was significantly worse than at least one other algorithm.

	Accuracy	AUROC	F_1	Precision	Recall	G-Mean	Specificity
LR							
RUS	2.35	2.60	2.40	2.38	2.87	2.42	2.39
KMUS	2.10	2.36	2.26	2.22	2.77	2.26	2.14
SBC	3.93	3.28	3.56	3.82	**1.13**	3.30	3.93
RRUS	**1.62**	**1.76**	**1.78**	**1.58**	~~3.23~~	**2.02**	**1.54**
3-NN							
RUS	2.86	3.02	2.82	2.82	2.69	2.64	2.75
KMUS	**1.42**	**1.60**	**1.74**	**1.48**	3.16	2.08	**1.34**
SBC	3.82	3.14	3.48	3.84	**1.24**	3.22	4.00
RRUS	1.90	~~2.24~~	1.96	1.86	~~2.91~~	**2.06**	~~1.91~~
DT							
RUS	2.00	2.26	2.16	1.98	2.91	2.12	1.98
KMUS	2.88	2.82	2.86	2.90	2.37	2.90	2.87
SBC	3.58	3.02	3.22	3.54	**1.49**	3.16	3.62
RRUS	**1.54**	**1.90**	**1.76**	**1.58**	~~3.23~~	**1.82**	**1.53**
SVM							
RUS	2.76	2.90	2.78	2.76	2.56	2.68	2.65
KMUS	**1.73**	2.21	2.23	**1.66**	3.71	2.83	**1.54**
SBC	3.66	3.38	3.38	3.62	**1.68**	3.24	3.66
RRUS	1.85	**1.51**	**1.61**	1.96	2.05	**1.25**	2.15
RF							
RUS	2.18	2.21	2.18	2.12	2.88	2.08	2.14
KMUS	2.87	2.83	2.90	2.72	2.43	3.02	2.78
SBC	3.57	3.19	3.38	3.66	**1.44**	3.20	3.66
RRUS	**1.38**	**1.77**	**1.54**	**1.50**	~~3.25~~	**1.70**	**1.42**
GBM							
RUS	2.12	2.36	2.00	2.10	2.80	2.06	2.18
KMUS	2.80	2.82	2.92	2.70	2.41	2.98	2.68
SBC	3.62	3.08	3.48	3.58	**1.56**	3.22	3.68
RRUS	**1.46**	**1.74**	**1.60**	**1.62**	~~3.23~~	**1.74**	**1.46**

Table 4.2: Number of times RRUS had the 1st, 2nd, 3rd, or 4th best mean rank for each classifier. Each row must add up to 7 since there are 7 performance metrics.

	1	2	3	4
LR	6	0	0	1
3-NN	1	5	1	0
DT	6	0	0	1
SVM	3	4	0	0
RF	6	0	0	1
GBM	6	0	0	1

Table 4.3: Number of times RRUS had the 1st, 2nd, 3rd, or 4th best mean rank for each performance metric. Each row must add up to 6 since there are 6 classifiers. Values in bold mean that RRUS achieved the 1st best mean rank for all classifiers.

	1	2	3	4
Accuracy	4	2	0	0
AUROC	5	1	0	0
F_1	5	1	0	0
Precision	4	2	0	0
Recall	0	1	1	4
G-Mean	**6**	0	0	0
Specificity	4	2	0	0

question, we analyse several plots comparing the performance of RRUS versus RUS and of RRUS versus KMUS on all data sets. Since there are many combinations of classifier and performance metric, we focus our attention on the G-Mean metric. The plots contain for all data sets the performance of RRUS on the y-axis and the performance of either RUS or KMUS on the x-axis. If performance of RRUS was greater than performance of RUS or KMUS, the name of the data set is plotted in green, whereas if performance of RRUS was inferior the name of the data set is plotted in red. In case of a tie in performance, the name of the data set is plotted in blue. For completeness, we also compare RRUS against SBC.

Figures 4.5, 4.6, 4.7, 4.8, 4.9, and 4.10 show the comparisons of RRUS versus RUS. For the LR and 3-NN classifiers G-Mean performance increased in most data sets, although the gains were modest. Nevertheless, for LR the data sets glass1 and yeast-0-3-5-9_vs_7-8 increased from 58.50% to 62.67% and from 66.42% to 69.83%, respectively, and for 3-NN the data set Blood T. S. C. increased from 62.94% to 67.45%. DT, on the other hand, benefited from RRUS in most data sets and in several of them the gains were moderate. The highest gains were for the Parkinsons and yeast-0-3-5-9_vs_7-8 data sets, which increased from 74.93% to 79.58% and from 60.01% to 66.70%, respectively. Many big performance gains happened for the SVM classifier, where the two greatest gains occurred for the oil and PC1" data sets, from 20.72% to 74.03% and from 48.43% to 69.22%, respectively. For the RF and GBM classifiers, most data sets benefited from RRUS and the gains were moderate. For instance, for RF the PC1" and the yeast-0-5-6-7-9_vs_4 data sets increased from 74.69% to 77.34% and from 73.62% to 77.19%, respectively, and for GBM the yeast-0-5-6-7-9_vs_4 increased from 75.09% to 78.54%.

Figures 4.11, 4.12, 4.13, 4.14, 4.15, and 4.16 show the comparisons of RRUS versus KMUS. For LR most data sets benefited from RRUS and the gains were modest. The biggest gain happened for the compustat data set, from 70.29% to 73.80%. For 3-NN, even though majority of the data sets did not benefit from RRUS, several of the data sets that did had moderate performance gains. For instance, the MC1" and MC2" data sets, from 56.51% to 62.83% and from 53.30% to 59.77%, respectively. For the remaining classifiers, DT, SVM, RF, and GBM,

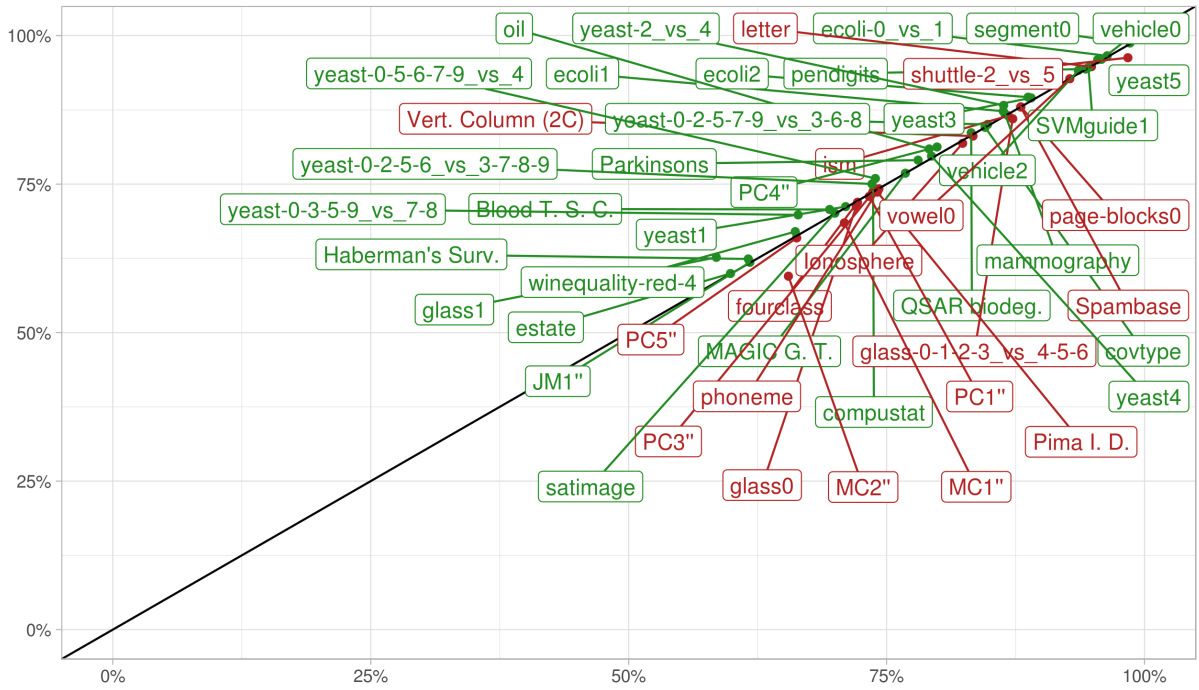

Figure 4.5: Comparison of the performance of RRUS (y-axis) versus RUS (x-axis), with LR as base classifier, according to G-Mean metric.

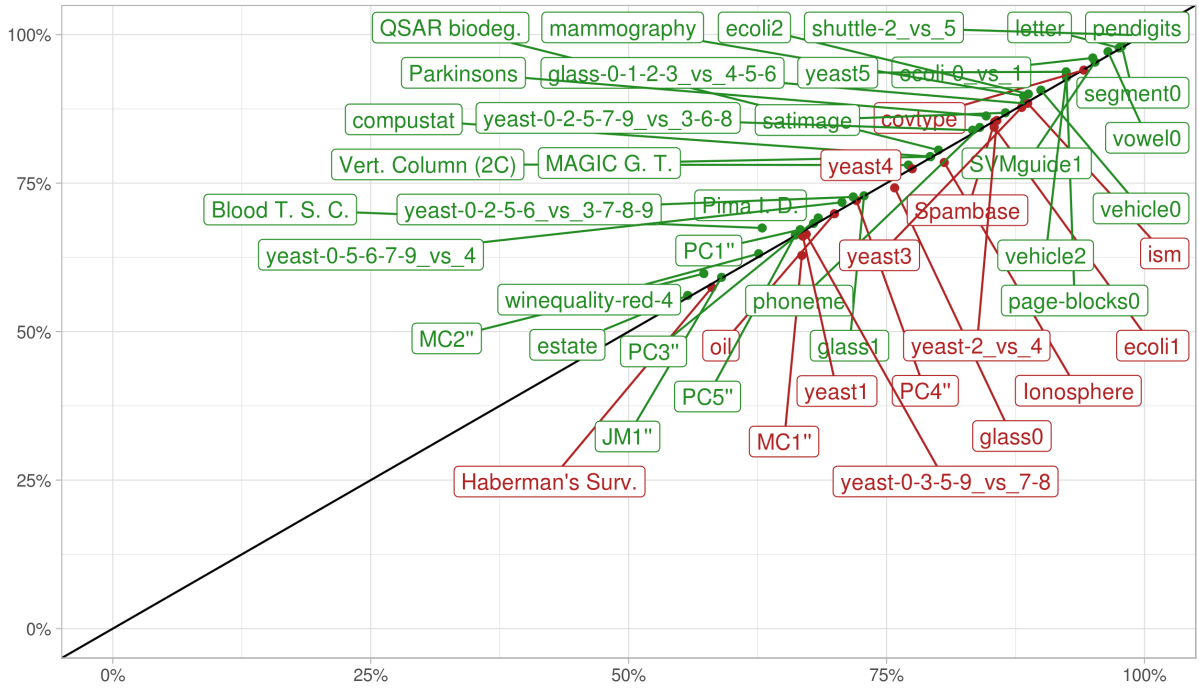

Figure 4.6: Comparison of the performance of RRUS (y-axis) versus RUS (x-axis), with 3-NN as base classifier, according to G-Mean metric.

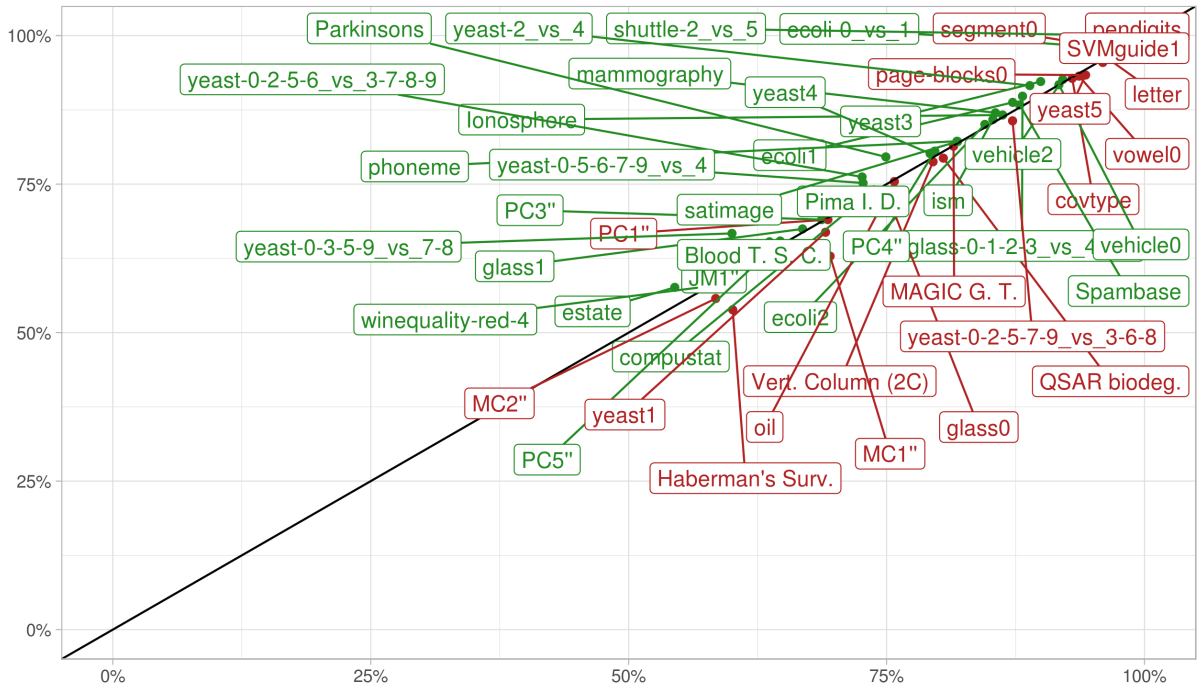

Figure 4.7: Comparison of the performance of RRUS (y-axis) versus RUS (x-axis), with DT as base classifier, according to G-Mean metric.

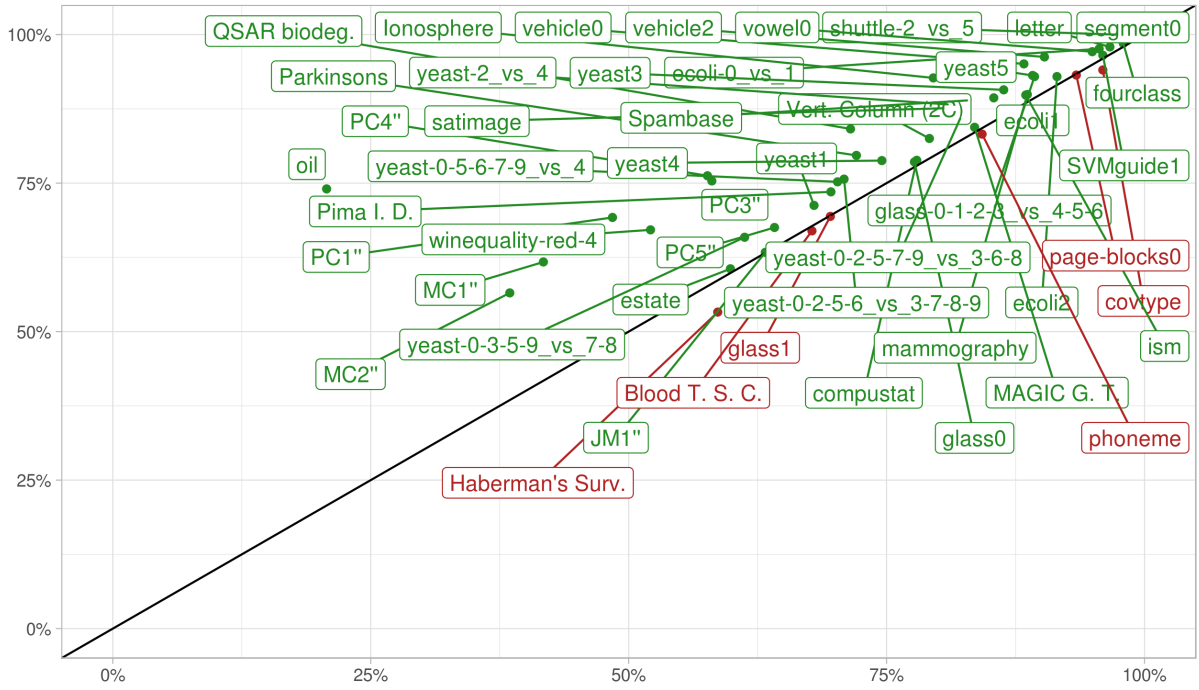

Figure 4.8: Comparison of the performance of RRUS (y-axis) versus RUS (x-axis), with SVM as base classifier, according to G-Mean metric.

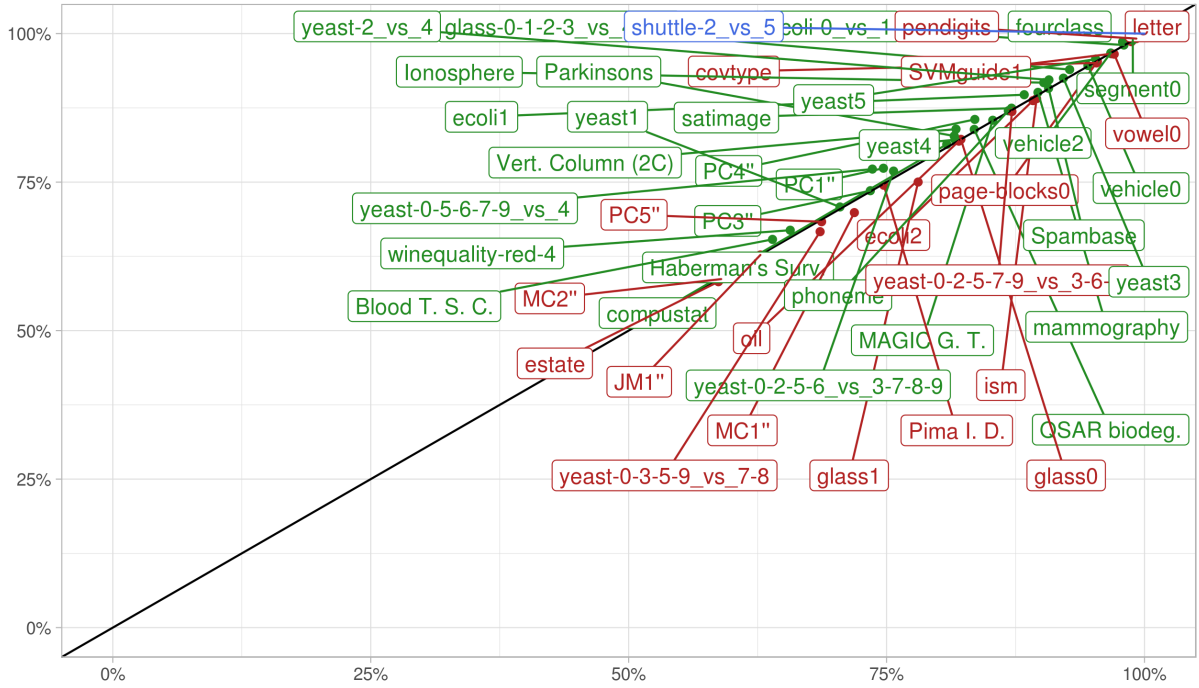

Figure 4.9: Comparison of the performance of RRUS (y-axis) versus RUS (x-axis), with RF as base classifier, according to G-Mean metric.

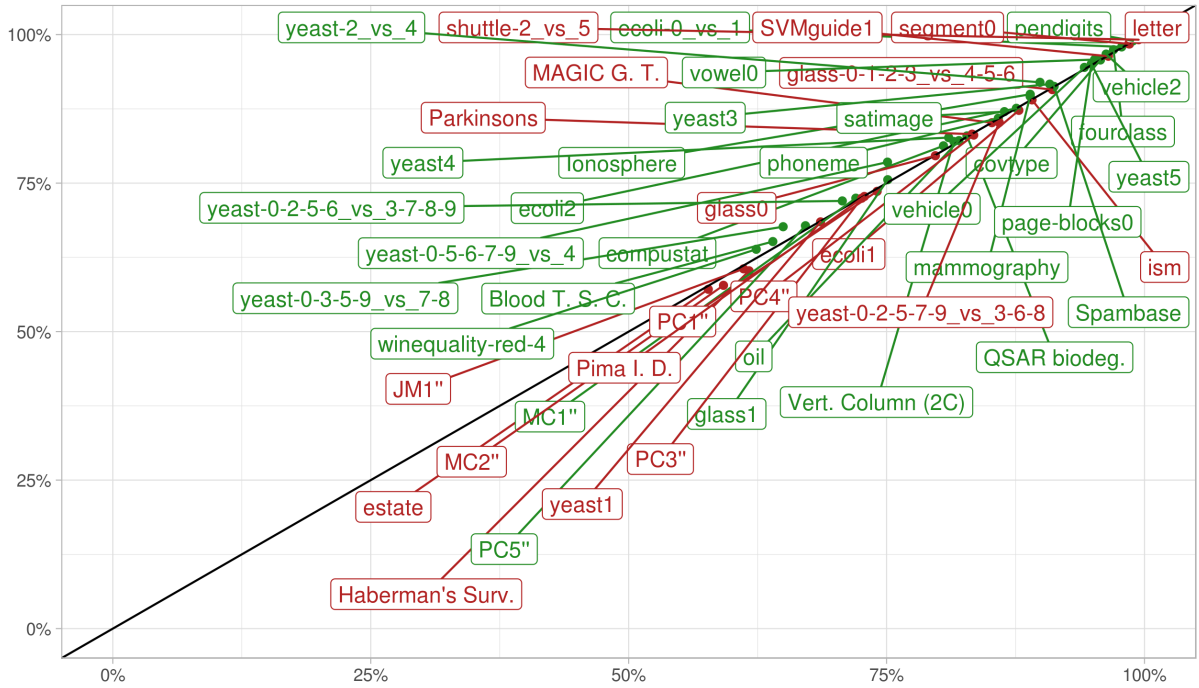

Figure 4.10: Comparison of the performance of RRUS (y-axis) versus RUS (x-axis), with GBM as base classifier, according to G-Mean metric.

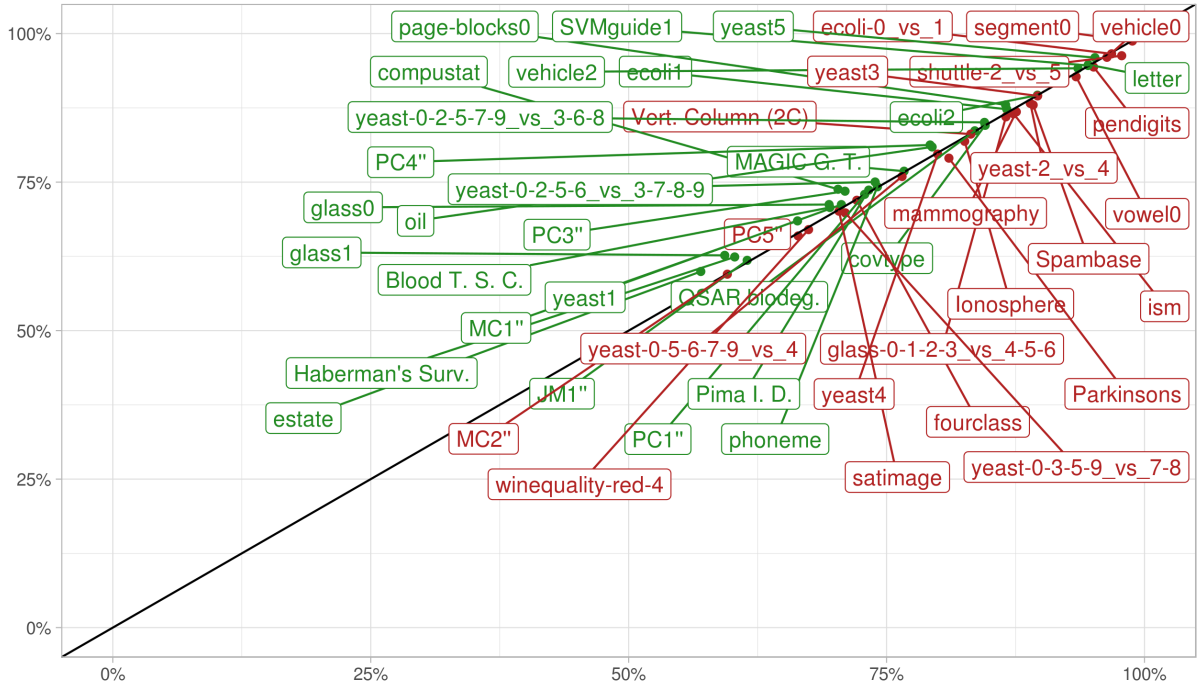

Figure 4.11: Comparison of the performance of RRUS (y-axis) versus KMUS (x-axis), with LR as base classifier, according to G-Mean metric.

the vast majority of the data sets benefited from RRUS, specially the SVM, where only 5 data sets did not increase performance over KMUS. The biggest increase in performance for DT was the JM1" data set, from 42.96% to 60.98%, for SVM was the oil data set, from 18.40% to 74.03%, for RF was the QSAR biodeg. data set, from 38.78% to 83.87%, and for GBM was the JM1" data set, from 37.17% to 60.57%.

Figures 4.17, 4.18, 4.19, 4.20, 4.21, and 4.22 show the comparisons of RRUS versus SBC. For the 3-NN and GBM classifiers RRUS had a better performance for most data sets and and the gains were moderate. The biggest gain in performance for the 3-NN classifier happened for the covtype data set which increased from 78.02% to 94.03%, and for the GBM classifier was the spambase data set, which increased from 76.64% to 91.28%. In the case of LR, DT, and RF the gains in performance were high. In particular, the biggest gain among these 3 classifiers happened for the DT classifier and the estate data set, where SBC had a mean performance of 18.66% and RRUS had a mean performance of 57.62%. For the SVM classifier, the gains were in general very high and only the Haberman's Survival data set had a better performance using SBC rather than RRUS. The highest gain in performance occurred for the oil data set, from 10.38% to 74.03%.

4.6 Summary

In this chapter we proposed RRUS, an under-sampling algorithm that selects a subset of the majority class that more appropriately represents the majority class. This is achieved by

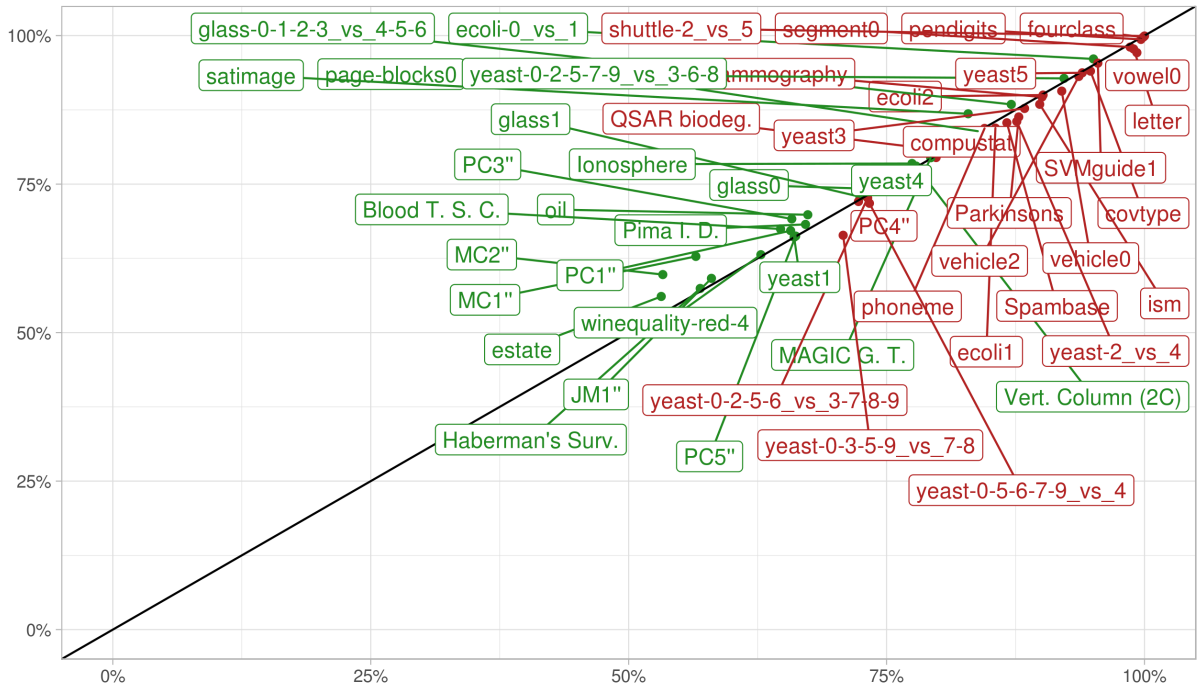

Figure 4.12: Comparison of the performance of RRUS (y-axis) versus KMUS (x-axis), with 3-NN as base classifier, according to G-Mean metric.

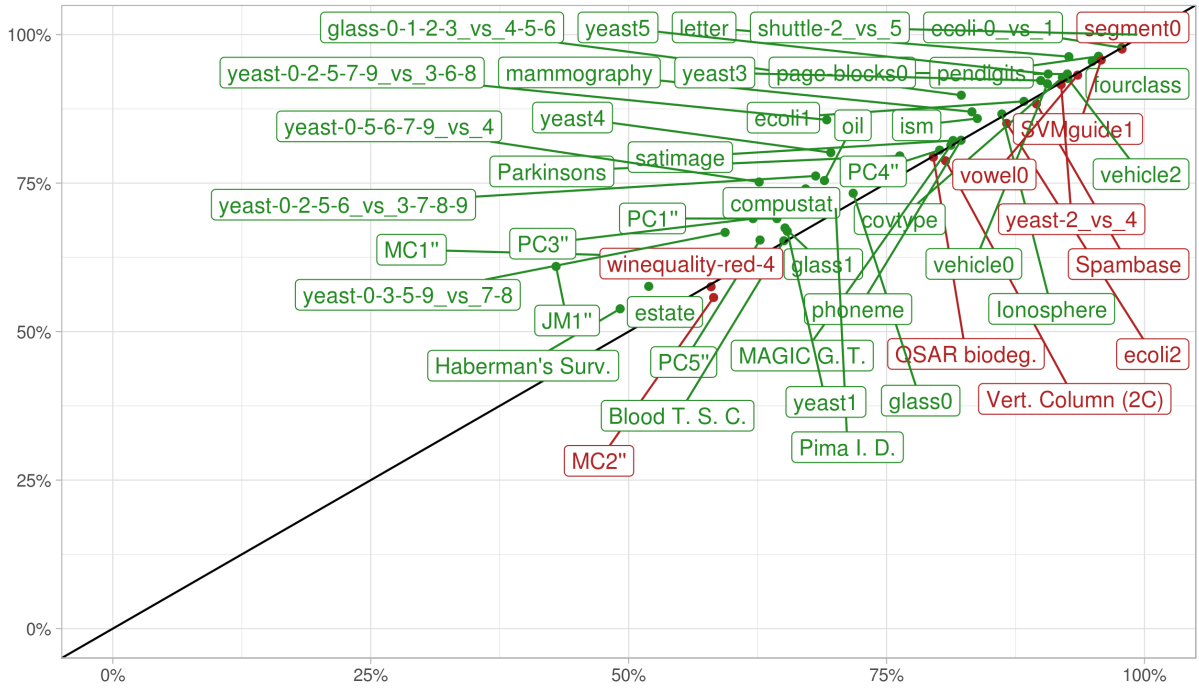

Figure 4.13: Comparison of the performance of RRUS (y-axis) versus KMUS (x-axis), with DT as base classifier, according to G-Mean metric.

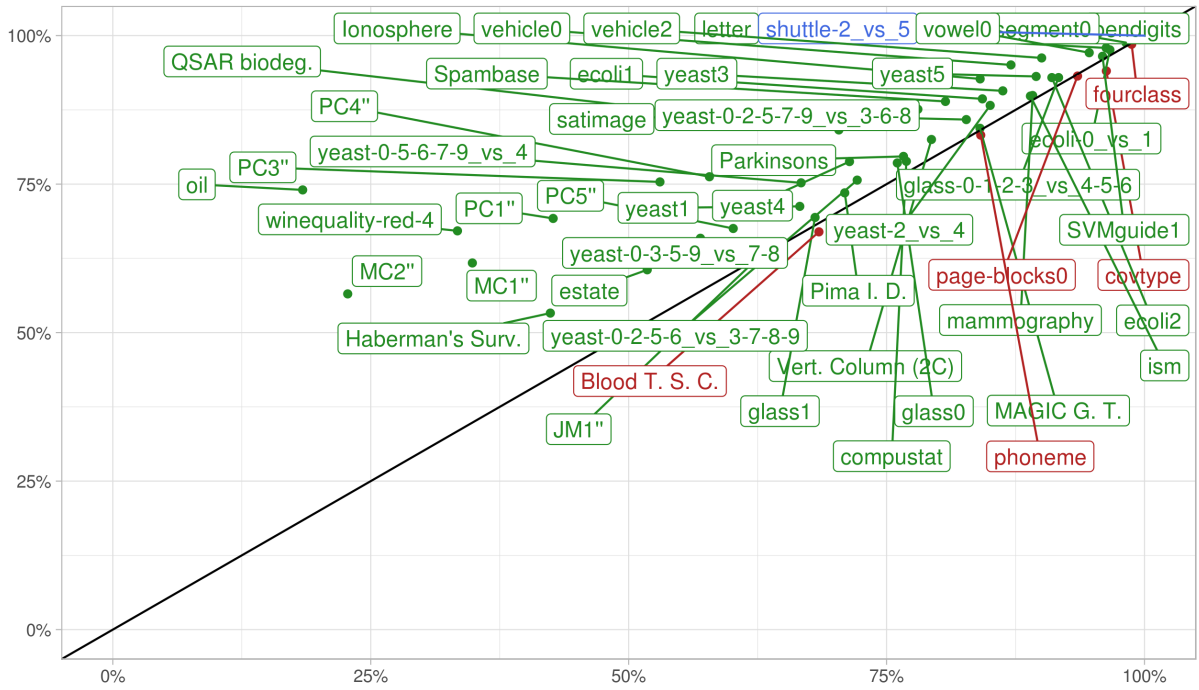

Figure 4.14: Comparison of the performance of RRUS (y-axis) versus KMUS (x-axis), with SVM as base classifier, according to G-Mean metric.

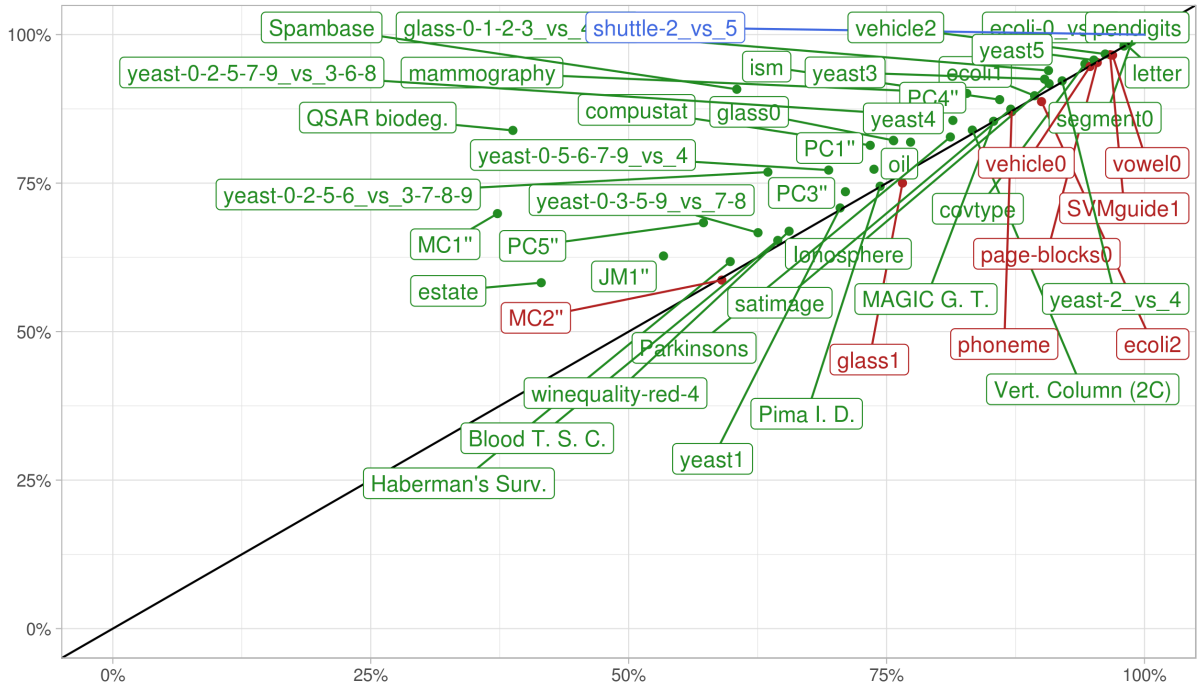

Figure 4.15: Comparison of the performance of RRUS (y-axis) versus KMUS (x-axis), with RF as base classifier, according to G-Mean metric.

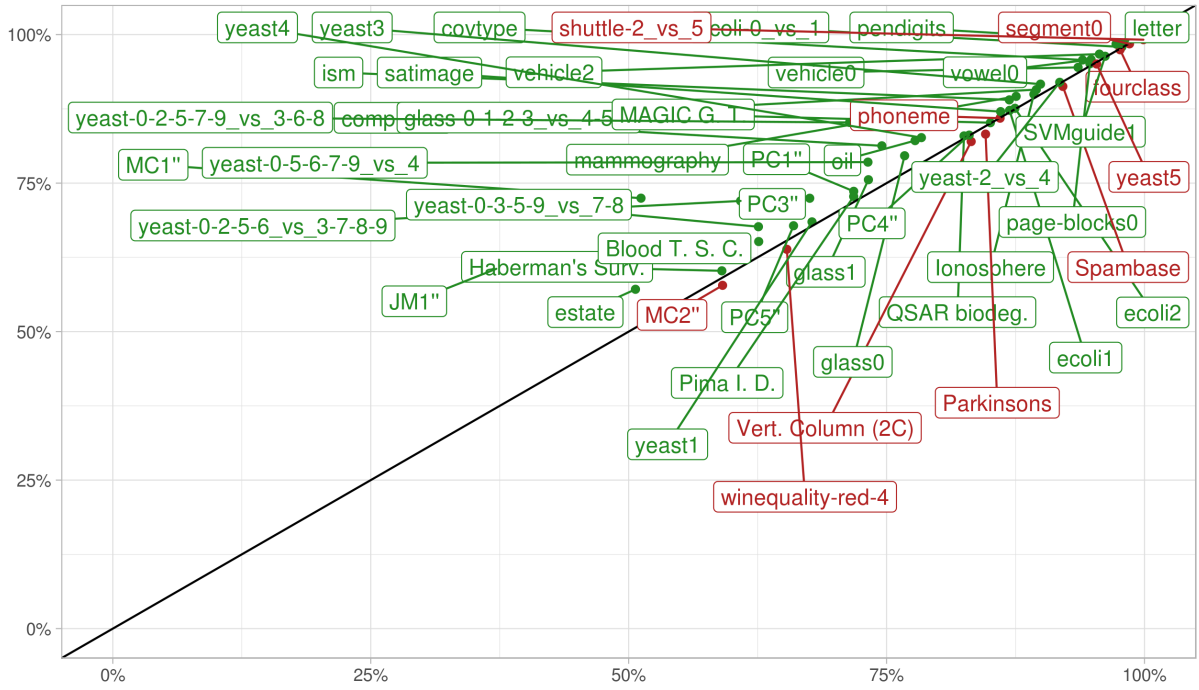

Figure 4.16: Comparison of the performance of RRUS (y-axis) versus KMUS (x-axis), with GBM as base classifier, according to G-Mean metric.

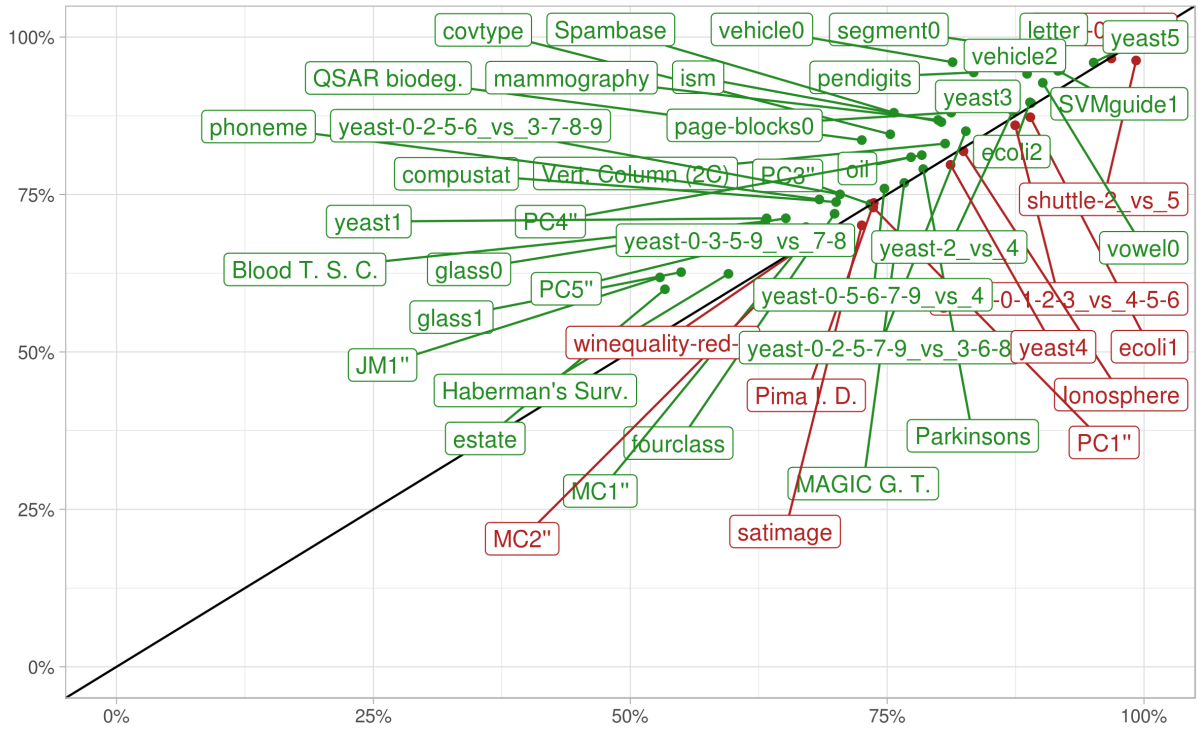

Figure 4.17: Comparison of the performance of RRUS (y-axis) versus SBC (x-axis), with LR as base classifier, according to G-Mean metric.

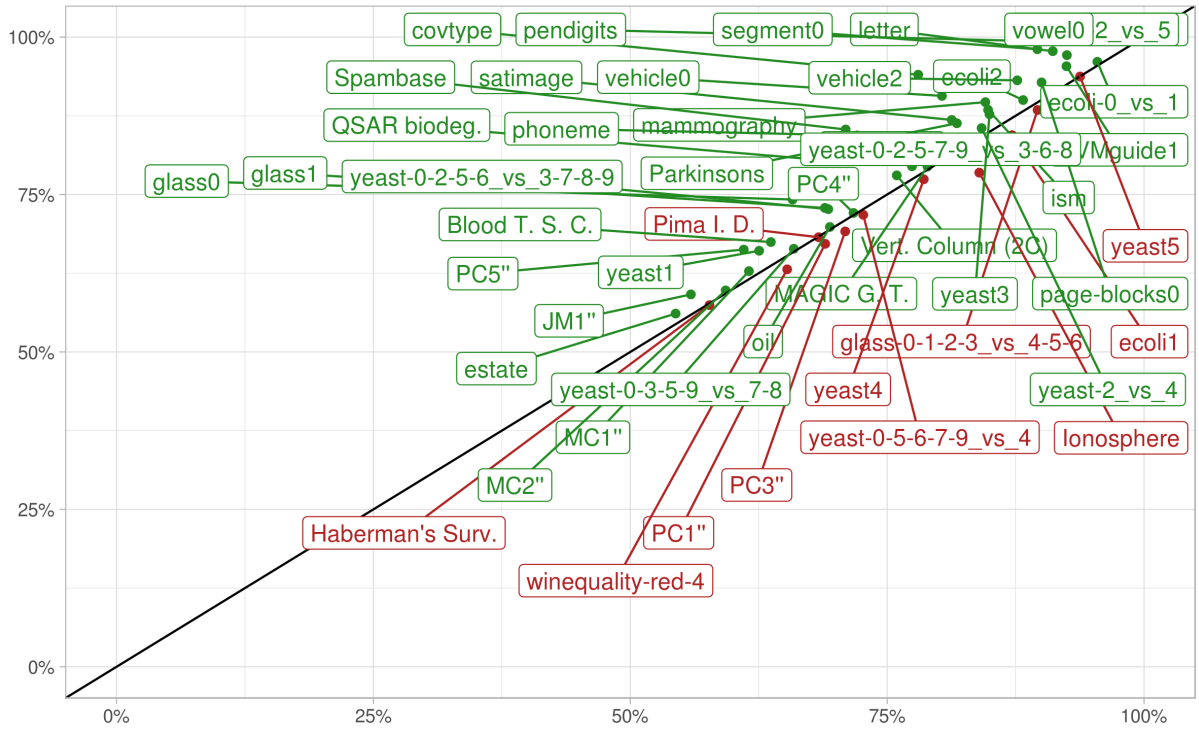

Figure 4.18: Comparison of the performance of RRUS (y-axis) versus SBC (x-axis), with 3-NN as base classifier, according to G-Mean metric.

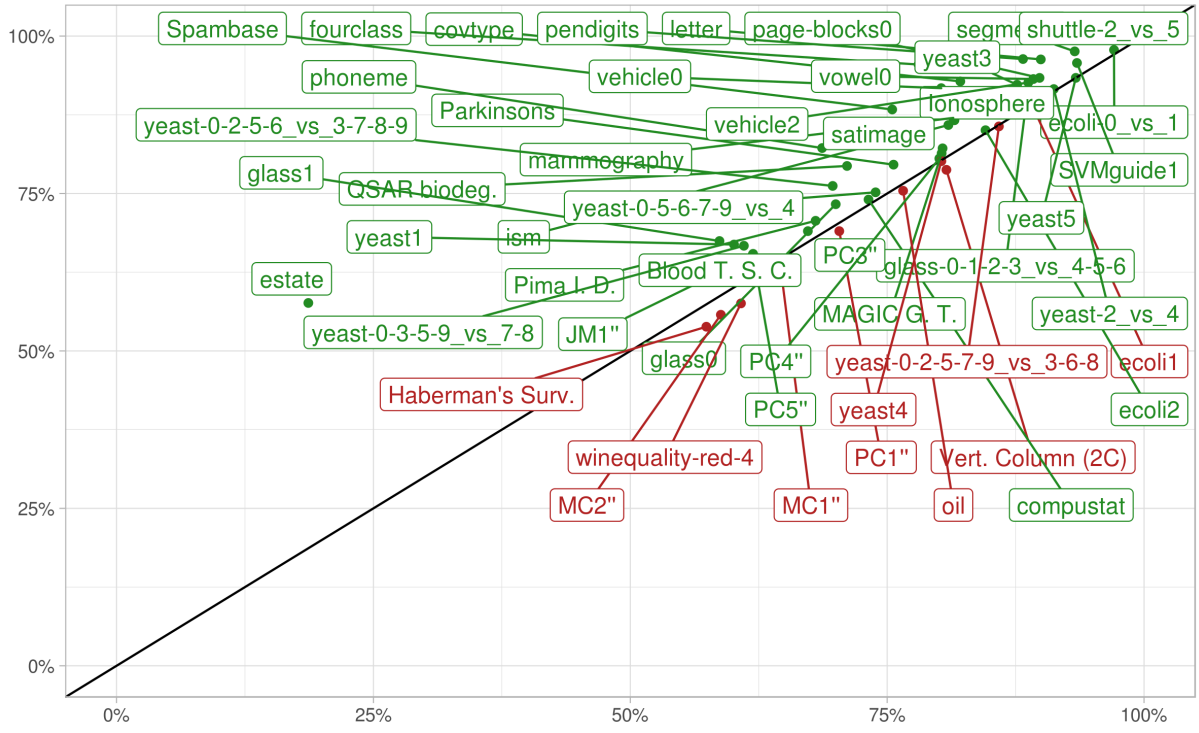

Figure 4.19: Comparison of the performance of RRUS (y-axis) versus SBC (x-axis), with DT as base classifier, according to G-Mean metric.

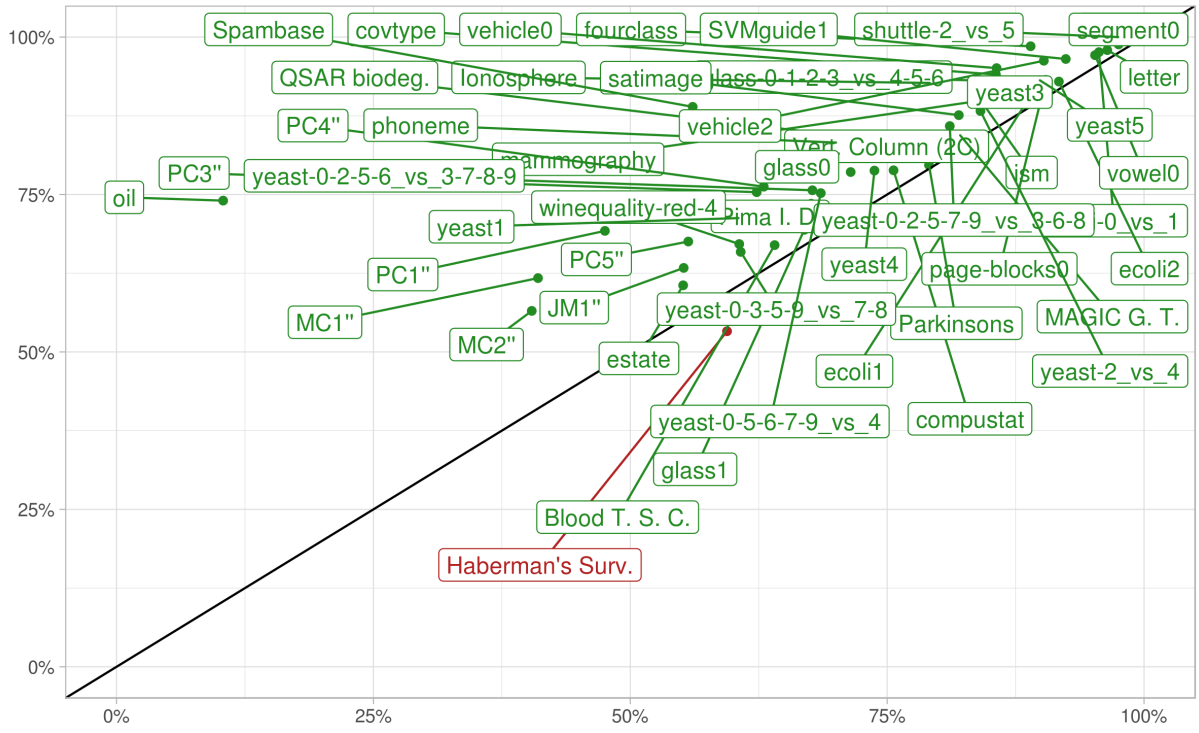

Figure 4.20: Comparison of the performance of RRUS (y-axis) versus SBC (x-axis), with SVM as base classifier, according to G-Mean metric.

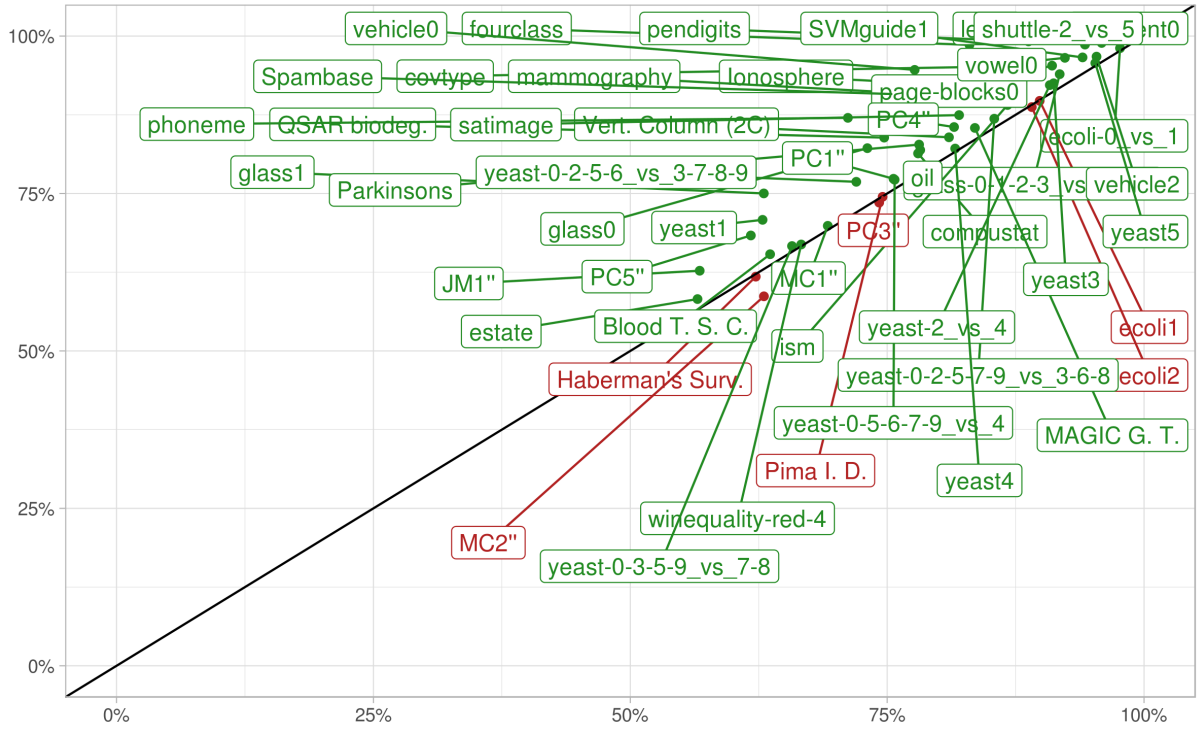

Figure 4.21: Comparison of the performance of RRUS (y-axis) versus SBC (x-axis), with RF as base classifier, according to G-Mean metric.

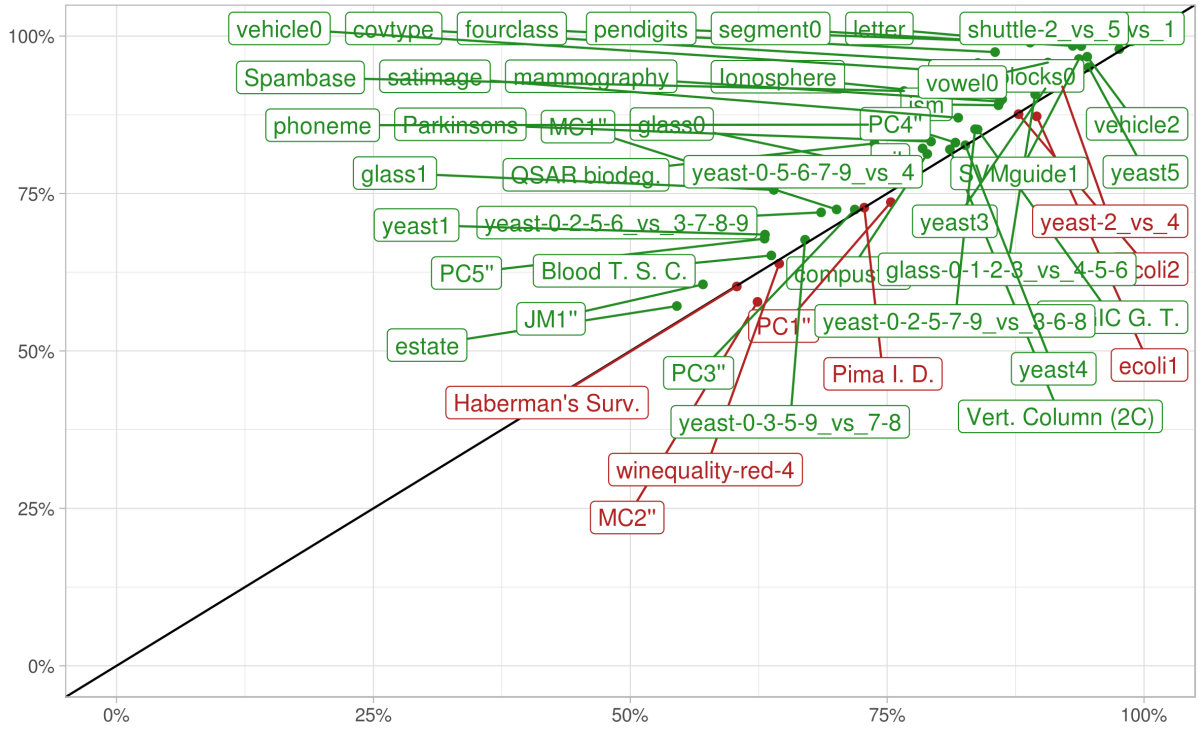

Figure 4.22: Comparison of the performance of RRUS (y-axis) versus SBC (x-axis), with GBM as base classifier, according to G-Mean metric.

repeatedly sampling random subsets from the majority class and selecting the subset that has its centroid closer to the majority class' centroid. To account for multimodal distributions, the majority class is first clustered using the k-Means algorithm and the aforementioned procedure is applied to each cluster. The disadvantages of RRUS over RUS are the increased execution time and the loss of generality of RUS (computation of the centroid of a set of examples requires numerical data).

The experimentation conducted to assess RRUS' benefits was comprehensive, and compared RRUS to the RUS, KMUS, and SBC under-sampling algorithms. All under-sampling algorithms were applied to 50 imbalanced data sets, which in turn were used to train 6 distinct classifiers, and their performance was evaluated on 7 performance metrics. The statistical significance of the results were tested using the *Friedman Aligned Ranks* test and the *Finner* post hoc test at the 5% significance level. RRUS consistently ranked best for most combinations of classifier and performance metric, being frequently significantly better than KMUS and SBC, and many times significantly better than RUS. Even though RRUS did not perform well for the Recall metric, the Precision/Recall trade-off characterised by the F_1 metric did benefit of RRUS in most cases.

To analyse the performance gains attained by RRUS against RUS, KMUS, and SBC, plots of the G-Mean metric performance were analysed for all classifiers. When compared against RUS, RRUS increased performance considerably in a number of data sets for the SVM classifier and had moderate to modest gains on the other classifiers. When compared to KMUS,

RRUS attained big performance increases in many data sets for the DT, SVM, RF, and GBM classifiers, and moderate gains for the LR and 3-NN classifiers. Finally, when compared against the SBC classifier, RRUS had very high gains for the SVM classifier, high gains for the LR, DT, and RF classifiers, and moderate gains for the 3-NN and GBM classifiers.

5

k-INFLUENTIAL NEIGHBOURHOOD FOR OVER-SAMPLING

In this chapter, we propose the *k*-Influential Neighbourhood for Over-Sampling (*k*-INOS) algorithm to enhance robustness of over-sampling algorithms to noisy examples present in the minority class. *k*-INOS first removes instances likely to be noise in the minority class, then applies an over-sampling algorithm to the under-sampled data set, and, finally, returns the removed minority examples back to the over-sampled data set. The removed minority examples, therefore, are not used to augment the minority class but are put back into the data set (after over-sampling) to avoid possible loss of genuine minority cases. A thorough experimentation was conducted, including 50 data sets, 6 classifiers, 7 over-sampling algorithms, and 7 performance metrics, to assess the usefulness of *k*-INOS. The experimentation results showed that *k*-INOS was able to increase Accuracy, F_1, Precision, and Specificity performances for most classifiers and over-sampling algorithms. In fact, the weak classifiers used experienced the most significant improvements. Positive results were also observed for the AUROC metric and strong classifiers, although with only part of them being statistically significant in this case.

In Chapter 6 two independent analysis of *k*-INOS are conducted. The first is a study of the hyperparameters of *k*-INOS that aims to understand their importance and provide suggestions on which values to use in practice. The second is an extended analysis of the results in this chapter, carried out to mine particular features and induce rules from the data sets to explain improvements in performance attained by *k*-INOS.

The rest of this chapter is organised as follows. Section 5.1 gives an overview of related works and highlights how the proposed algorithm fits into current literature. Section 5.2 reviews the concept of neighbourhood of influence and explains the proposed method. Section 5.3 describes the specific experimentation settings employed to assess the usefulness of *k*-INOS. Section 5.4 presents the results achieved and discusses the main findings. Finally, section 5.5 summarises the chapter and draws the conclusions.

5.1 Related Work

Removal of examples from the minority class prior to over-sampling is rarely seen in the literature. As minority examples are scarce, most attempts have aimed at the post processing

stage, removing noisy examples potentially introduced by over-sampling algorithms. Again, all articles mentioned in this section have been reviewed in Chapter 2, but we briefly describe them again and include information about the experimentation conducted by the authors.

(BATISTA; PRATI; MONARD, 2004) proposed the use of Wilson's Edited Nearest Neighbour Rule (ENN) (WILSON, 1972) or Tomek Link (TL) (TOMEK, 1976) as a cleaning step after over-sampling with SMOTE. ENN removes the examples misclassified by their *k*-nearest neighbours while TLs removes the examples that belong to different classes and are the nearest neighbours of each other. The authors assessed the proposed methods, SMOTE + ENN and SMOTE + TLs, on 10 data sets using a C4.5 Decision Tree as base classifier and AUROC as the performance metric, and argued that both methods performed better than SMOTE alone.

(SÁEZ et al., 2015) proposed the use of an iterative ensemble based noise filter to remove noisy examples after over-sampling with SMOTE. The aim was to remove noisy examples already present in the data and noisy examples introduced by SMOTE. The experimentation employed to assess the proposed method, SMOTE-IPF, comprised 9 data sets, C4.5 Decision Tree as base classifier, and AUROC as the performance metric. SMOTE-IPF results outperformed SMOTE, SMOTE + ENN, and SMOTE + TLs. The authors also reported testing SMOTE-IPF with other base classifiers and observed that SMOTE-IPF worked better with weak classifiers.

Recently, (RIVERA, 2017) proposed Noise Reduction A Priori Synthetic Over-Sampling (NRAS), which removes noisy examples from the minority class prior to over-sampling. To each example in the minority class a propensity score is assigned, or a probability of belonging to the minority class, and examples that have propensity scores below a predefined threshold are removed from the data set. An experimentation was conducted to assess NRAS, including 41 data sets, 4 base classifiers, and 3 performance metrics. NRAS was compared to several over-sampling algorithms and consistently attained a higher average rank for Recall and G-Mean metrics.

As discussed here and in Chapter 2, many algorithms were created to address the main drawback of SMOTE, which is the generation of new examples based solely on the minority examples (without analysing local context). Most adaptations add a post-processing step to clean up noisy examples introduced by SMOTE and very few add a pre-processing step. *k*-INOS follows an approach similar to NRAS with the extra benefit of being general enough to be used by any over-sampling algorithm.

5.2 *k*-INOS

k-INOS is a general wrapper for over-sampling algorithms grounded on the concept of neighbourhood of influence, as defined by (HA; LEE, 2016). The main purpose of *k*-INOS is to enhance the robustness of over-sampling algorithms to noisy examples in the minority class. It achieves that by removing examples from the minority class likely to be noise, prior to over-sampling. As minority examples are usually scarce and the removed examples could be

genuine, the removed instances are added back to the over-sampled data.

The neighbourhood of influence of an example x is defined in terms of the nearest neighbours and reverse nearest neighbours of x. In the following definitions, consider a data set D and an example $x \in D$:

Definition 1. (k-nearest neighbours of x): The k points from $D \setminus \{x\}$ closest to x are called the k-nearest neighbours of x and are represented by k-NN(x).

Definition 2. (reverse k-nearest neighbours of x): The points from $D \setminus \{x\}$ that have x as one of their k-NNs are called the reverse k-nearest neighbours of x and are represented by k-RNN(x).

Definition 3. (k-influential neighbourhood of x):

$$k\text{-IN}(x) = \{x\} \cup k\text{-NN}(x) \cup k\text{-RNN}(x)$$

The definition of k-IN(x) can be understood in terms of two regions in feature space. There is the region where example x has direct influence ($\{x\} \cup k$-NN(x)) and the region where x has indirect influence (k-RNN(x)). The size of k-IN(x) is mainly determined by k-RNN(x) as the size of $\{x\} \cup k$-NN(x) is always $k+1$. Hence, a large k-IN(x) means example x is located in a dense region in feature space.

As we are interested in the influence of an example inside its class, we define the modified neighbourhood of influence by removing from the neighbourhood of influence of an example x all the examples from the opposite class. In particular, the modified neighbourhood of influence is used by k-INOS to remove the unwanted examples from the minority class.

Definition 4. (modified k-influential neighbourhood of x): For a two-class problem, the modified k-influential neighbourhood of x is defined as the subset of k-IN(x) that contains only examples from the same class of x.

Definition 4 is necessary because k-INOS uses the size of the neighbourhood of influence of examples from the minority class to determine which examples are likely to be noise. A minority example x surrounded by many majority examples is likely to be noise and have a large neighbourhood of influence. Therefore, by filtering out from the neighbourhood of influence of x all examples from the majority class, k-INOS is able to remove examples from the minority class that are likely to be noise based on the size of their modified neighbourhood of influence.

k-INOS has three main steps: pre-processing, over-sampling, and post-processing. First, for each minority example, the modified neighbourhood of influence is computed and the minority examples with a modified neighbourhood of influence smaller than a pre-defined threshold τ are removed. This step aims to filter out minority examples that may negatively influence the over-sampling process. Second, the under-sampled data set is over-sampled. And third, in the post-processing step, the minority examples removed during pre-processing are added back to the over-sampled data set. This last step is performed to avoid losing information, since the

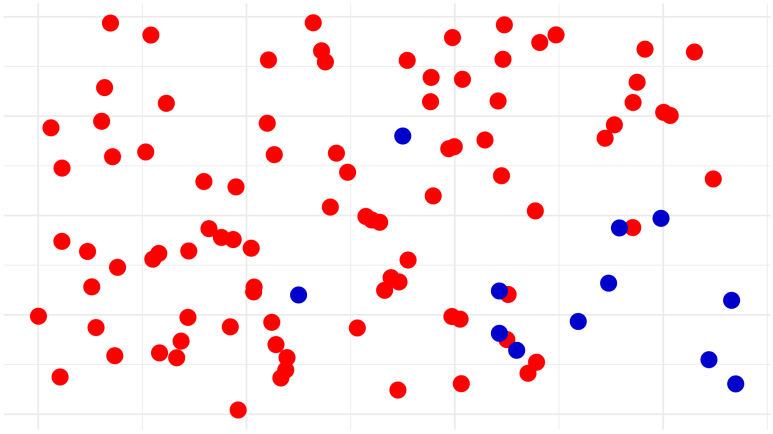

Figure 5.1: Definition 1 applied to an imbalanced data set containing noise. As k-NN(x) has k examples for every example x, all examples have the same importance.

removed examples could correspond to genuine data and the minority class usually has few examples. A full description of k-INOS is given in Algorithm 2.

The behaviour of k-INOS is controlled by two hyperparameters, k and τ. k is the number of neighbours employed to compute the modified neighbourhood of influence and controls how far k-INOS should consider the influence of an example in feature space. τ is a threshold used to remove examples from the minority class that have modified neighbourhoods of influence smaller than τ. The higher the value of τ the greater the number of examples removed.

Example 3. To illustrate Definitions 1-4, Figures 5.1, 5.2, 5.3, and 5.4 plot heat maps of the "importance" of examples in a two-dimensional imbalanced data set, where the importance is measured proportionally to the size of the associated set. The more intense the colour and the larger the point, the greater the importance associated with the example. The effect of Definition 1 is illustrated in Figure 5.1. Since the set of k-NNs of each example has k examples, all examples have the same importance. In Figure 5.2, however, an uneven distribution of importance appears after application of Definition 2. Considering only the k-RNNs, the most important examples are the ones located in dense regions of feature space. Figure 5.3 shows the effect of Definition 3. k-IN is able to concentrate the importance in minority examples far from majority examples, but the majority examples from k-RNN still impose some influence on each minority example. Finally, Figure 5.4 illustrates the effect of Definition 4, slightly modified to remove examples from the majority class instead of the opposite class. We do this because, in practice, k-INOS only computes the modified neighbourhood of influence for the examples of minority class. As it can be seen, the importance is now correctly concentrated on the minority examples far from the majority class.

Example 4. To exemplify the importance of selecting an appropriate subset of examples from the minority class before over-sampling, we contrast the behaviour of SMOTE with and without k-INOS in Figures 5.6 and 5.7. The original two-dimensional imbalanced data set is shown in

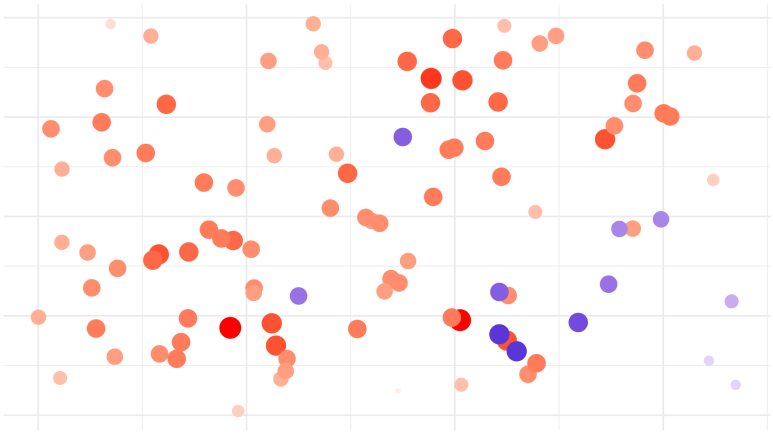

Figure 5.2: Definition 2 applied to an imbalanced data set containing noise. Now, the importance shifts to examples located in denser regions in feature space.

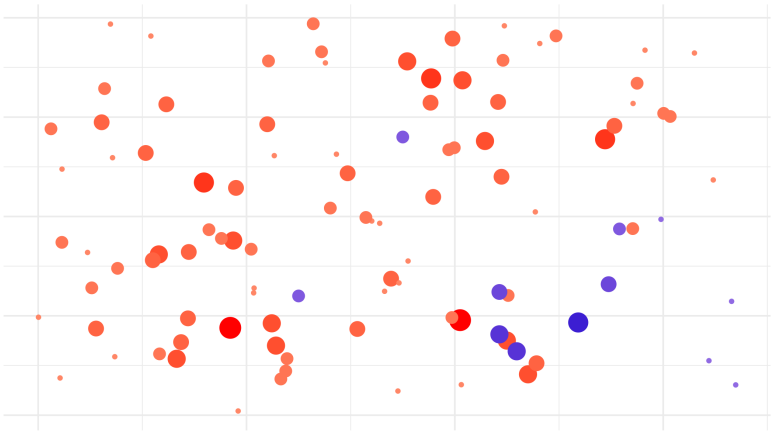

Figure 5.3: Definition 3 applied to an imbalanced data set containing noise. The importance starts to decrease in noisy minority examples, however it is still far from ideal.

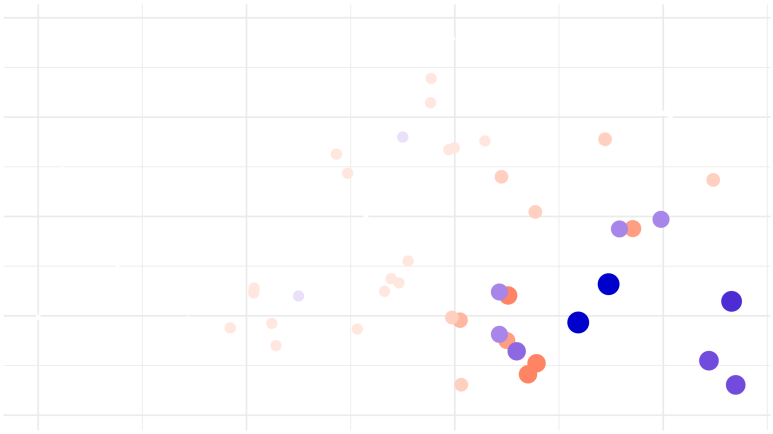

Figure 5.4: Definition 4 applied to an imbalanced data set containing noise. However, instead of removing examples from the opposite class, the examples from majority class are removed for both classes. This is done to highlight the behaviour of *k*-INOS, as it never computes the modified neighbourhood of influence for examples from the majority class. Notice that now noisy examples from the minority class have little importance.

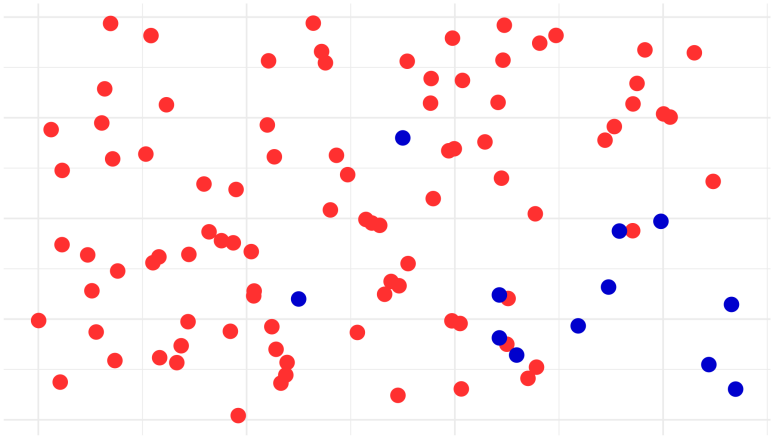

Figure 5.5: An imbalanced data set containing noisy examples.

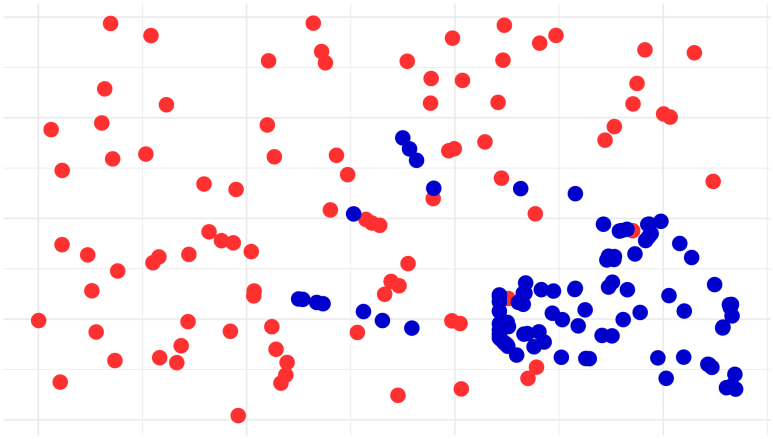

Figure 5.6: Over-sampling the data set in Figure 5.5 using SMOTE ($k = 5$).

Figure 5.5. As it can be seen, SMOTE synthesises new examples employing every instance already available in the minority class. This blindness to which pair of examples to use to synthesise new examples is the main drawback of SMOTE, as it can generate many noisy examples. On the other hand, by removing examples in the minority class likely to be noise, *k*-INOS provides a better data set for SMOTE to over-sample. In the employed data set, some isolated minority examples within the majority class region are used by SMOTE, which creates new minority examples in the majority class region, as seen in Figure 5.6. In Figure 5.7, with application of *k*-INOS, the isolated minority examples remain in the over-sampled data, however, they are not utilised by SMOTE to synthesise new examples, thus preventing the spread of noisy examples.

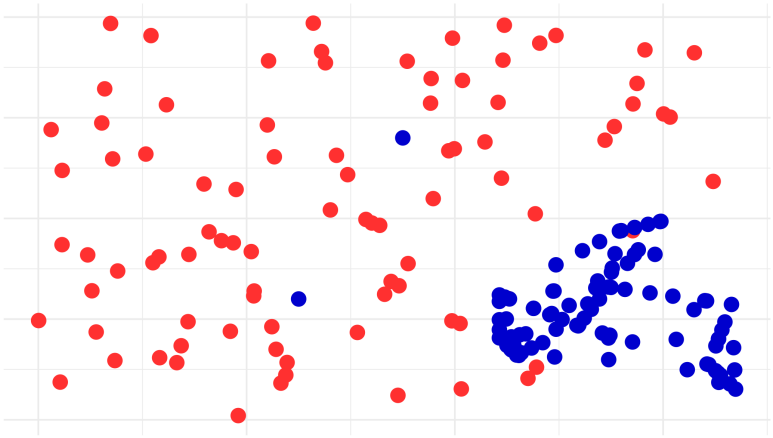

Figure 5.7: Over-sampling the data set in Figure 5.5 using SMOTE ($k = 5$) and k-INOS ($k = 7$ and $\tau = 2$).

Algorithm 2: k-INOS

Objective

Enhance the robustness of over-sampling algorithms to noise by removing examples from the minority class prior to over-sampling.

Input

- D: An imbalanced data set.
- k: Number of neighbours to compute the modified k-IN.
- τ: Threshold value to remove examples from the minority class.
- ϕ: An over-sampling algorithm.

Output

- $D*$: A more balanced version of D.

Algorithm

1. ***k*-IN**: For each example in the minority class compute its modified k-IN as in **Definition 4**.
2. **Pre-Process**: Remove the minority class examples that have a modified k-IN smaller than τ.
3. **Over-Sample**: Over-sample the pre-processed data using ϕ.
4. **Post-Process**: Return the removed examples to the final over-sampled data set $D*$.

5.3 Experimental Methodology

To assess the usefulness of wrapping over-sampling algorithms with k-INOS, we tested k-INOS on 7 over-sampling algorithms. The experimental framework utilised is described in Chapter 3 and includes 50 imbalanced data sets, 6 classifiers, and 7 performance metrics. To verify if k-INOS led to statistically significant improvements, the Wilcoxon signed-ranks test (WILCOXON, 1945) was employed as recommended in (DEMŠAR, 2006). A one-sided hypothesis test and significance level of 5% were employed.

As k-INOS works as a wrapper for over-sampling algorithms, it is important to thoroughly test it with many different methods. Hence, 7 over-sampling algorithms were selected for experimentation: ROS, SMOTE (CHAWLA et al., 2002), BDL-SMOTE-{1, 2} (HAN; WANG; MAO, 2005), ADASYN (HE et al., 2008), SL-SMOTE (BUNKHUMPORNPAT; SINAPIROMSARAN; LURSINSAP, 2009), and RWO (ZHANG; LI, 2014). Details about each of them can be found in Chapter 2.

All over-sampling algorithms were set to balance the distribution of examples. In the algorithms where hyperparameter k is used, this was set to 5. For BDL-SMOTE-{1, 2}, hyperparameter m was set to 11 and, for ADASYN, hyperparameter d_{th} was set to 0.75. All the above values are default values recommended by the authors, with the exception of hyperparameter m in BDL-SMOTE, where the suggested value is not precisely defined. Also, hyperparameter d_{th} in ADASYN, a threshold to decide whether ADASYN is used, is not particularly important in our case, since the ratio 1/IR of all data sets used in this dissertation are smaller than 0.75 (in this case ADASYN is always used).

For k-INOS, k and τ were set to 11 and 3, respectively, and a further examination of their influence is presented in Chapter 6. Note that most over-sampling algorithms have a hyperparameter k used to compute the nearest neighbours, however, in k-INOS, k is used to compute the neighbourhood of influence.

5.4 Results and Discussion

For each combination of performance metric, classifier, and over-sampling algorithm, Table 5.1 shows the results of the Wilcoxon signed-ranks test. There are five possible entries in the table: $\triangle$, ▲, $\triangledown$, ▼, and "-". ▲ means k-INOS increased performance in more data sets than it decreased and the improvement was statistically significant, while $\triangle$ has the same meaning but the result was not statistically significant. On the other hand, $\triangledown$ and ▼ represent the opposite situations and, in this case, k-INOS decreased the performance. "-" means a tie in the number of data sets with increases or decreases.

Accuracy, Precision, and Specificity had significant increases in performance for most combinations of classifier and over-sampling algorithm. The F_1 metric also experienced many improvements, with over 35% of them being significant. AUROC increased for many over-

Table 5.1: Results of the Wilcoxon signed-ranks test. Black arrows indicate statistically significant increase/decrease in performance whereas transparent arrows indicate increase/decrease but not statistically significant. "-" represents ties. BDL-SMOTE-1, BDL-SMOTE-2, and SL-SMOTE names were abbreviated to BDL-1, BDL-2, and SL, respectively.

	ROS	SMOTE	BDL-1	BDL-2	ADASYN	SL	RWO
Accuracy							
LR	-	▲	▲	▲	▼	▲	▲
3-NN	-	▲	▲	▲	▲	▲	▲
DT	▲	▲	▲	▲	▲	▲	▲
SVM	▲	▲	▲	▲	-	▲	▲
RF	△	▲	▲	▲	△	▽	△
GBM	▲	▲	▲	▲	△	▲	▽
AUROC							
LR	△	▼	▽	▽	▼	▽	▼
3-NN	▽	△	△	▲	△	△	▲
DT	▼	▽	▽	-	△	△	△
SVM	▼	▽	▼	▼	▼	▼	▼
RF	△	▽	▽	△	▼	▽	▽
GBM	△	△	△	△	▽	△	▽
F_1							
LR	-	▲	▲	▲	▽	▲	▲
3-NN	▽	▲	▲	▲	△	▲	▲
DT	▲	▲	△	-	▽	△	▲
SVM	▽	△	-	△	▽	△	△
RF	▼	▽	▽	-	▽	-	▲
GBM	△	△	△	△	▲	▲	▽
Precision							
LR	-	▲	▲	▲	▽	▲	▲
3-NN	▽	▲	▲	▲	▲	▲	▲
DT	▲	▲	▲	▲	▲	▲	▲
SVM	▲	▲	▲	▲	▲	▲	▲
RF	▲	▲	▲	▲	△	△	▲
GBM	▲	▲	▲	▲	△	▲	△
Recall							
LR	-	▼	▼	▼	▲	▼	▼
3-NN	▽	▼	▽	-	△	▽	▽
DT	▼	▼	▼	▼	▽	▽	▲
SVM	▼	▼	▼	▼	▼	▼	▼
RF	▼	▼	▼	▼	▽	▼	△
GBM	▽	▼	▼	▼	▲	▽	▽
G-Mean							
LR	-	▽	△	▽	-	▽	▽
3-NN	▽	▼	△	△	△	△	△
DT	▼	▼	▽	▼	▼	▽	▲
SVM	▼	▼	▼	▼	▼	▼	▼
RF	▼	▼	▼	▼	△	▽	△
GBM	▽	▼	▼	▽	▲	△	▽
Specificity							
LR	-	▲	▲	▲	▼	▲	▲
3-NN	△	▲	▲	▲	▲	▲	▲
DT	▲	▲	▲	▲	▲	▲	▲
SVM	▲	▲	▲	▲	△	▲	▲
RF	▲	▲	▲	▲	△	▲	▽
GBM	▲	▲	▲	▲	△	▲	▽

sampling algorithms when considering the 3-NN and GBM classifiers, but for other classifiers it mainly decreased. Recall and G-Mean mainly consisted of decreases in performance, with 84% and 69% of decreases, respectively.

Accuracy had significant increases in 76.2% (32 out of 42) of the combinations of classifier and over-sampling algorithm. DT had significant increases for all over-sampling algorithms, 3-NN and SVM had significant increases in 6 out of 7 over-sampling algorithms, and LR and GBM had significant increases in 5 out of 7 over-sampling algorithms. GBM and RF had 1 decrease in performance each. Regardless of the classifier, SMOTE, BDL-SMOTE-1, and BDL-SMOTE-2 had only significant increases. Only 1 significant decrease happened and it was for the combination LR/ADASYN.

Precision had significant increases in 83.3% (35 out of 42) of the combinations of classifier and over-sampling algorithm. DT and SVM had significant increases for all over-sampling algorithms. 3-NN had 6 out of 7 significant increases and 1 decrease whereas LR, GBM, and RF had 5 out of 7 significant increases each. Similarly to Accuracy, SMOTE, BDL-SMOTE-1, and BDL-SMOTE-2 had only significant increases. No significant decreases occurred for Precision.

AUROC performance was spread in the 5 possible outcomes. In 40.5% of the combinations (17 out of 42) performance increased (2 of them significantly), in 57.1% of them (24 out of 42) performance decreased (11 of them significantly) and, in only 1 case, there was no change. 3-NN had 6 increases out of 7, with 2 of them significant. GBM and DT had 5 and 3 out of 7 increases, respectively, although none of them significant. LR had 1 increase and SVM had only decreases with 6 of them significant.

Recall had significant decreases for over half the combinations of classifier and over-sampling algorithm. SVM had significant decreases for all the over-sampling algorithms. LR and RF had 5 significant decreases while 3-NN had only 1 significant decrease. The only significant increases occurred for the combinations LR/ADASYN, DT/RWO, and GBM/ADASYN, and non-significant increases occurred for the combinations RF/RWO and 3-NN/ADASYN.

$\boldsymbol{F_1}$, the harmonic mean between Precision and Recall, had 64.3% of increases (27 out of 42) and majority of them were significant. LR had 5 out of 7 increases and all of them were significant. 3-NN and GBM had 6 out of 7 increases where 5 and 2 of them were significant, respectively. DT had 5 out of 7 increases with 3 of them significant. SVM had 4 out of 7 increases while RF had 4 decreases. Irrespective of the classifier, RWO, SMOTE, and SL-SMOTE had 5 out of 6 increases, with 4, 3, and 3 of them significant, respectively.

Specificity had significant increases in 80.9% (34 out of 42) of the combinations of classifier and over-sampling algorithm. DT, 3-NN, and SVM had increases in performance for all over-sampling algorithms, with 7, 6, and 6 of them being significant, respectively. RF and GBM had 6 increases each, where 5 of them were significant, and LR had 5 increases, all of them significant. The only decreases happened for the combinations LR/ADASYN, RF/RWO, and GBM/RWO, and only the first one was significant.

Table 5.2: Summary of the results presented in Table 5.1. For each performance metric a count of the possible outcomes is shown. The percentages were rounded to the nearest integer, therefore the rows may not add up to 100%.

	▲	△	▼	▽	-
Accuracy	32 (76%)	4 (10%)	1 (2%)	2 (5%)	3 (7%)
AUROC	2 (5%)	15 (36%)	11 (26%)	13 (31%)	1 (2%)
F_1	16 (38%)	11 (26%)	1 (2%)	9 (21%)	5 (12%)
Precision	35 (83%)	4 (10%)	0 (0%)	2 (5%)	1 (2%)
Recall	3 (7%)	2 (5%)	25 (60%)	10 (24%)	2 (5%)
G-Mean	2 (5%)	9 (21%)	18 (43%)	11 (26%)	2 (5%)
Specificity	34 (81%)	4 (10%)	1 (2%)	2 (5%)	1 (2%)
Total	124 (42%)	49 (17%)	57 (19%)	49 (17%)	15 (5%)

G-Mean had a significant decrease in performance for 42.8% of the combinations of classifier and over-sampling algorithm. SVM had a significant decrease in performance for all over-sampling algorithms. RF and GBM had 5 decreases each, with 4 and 2 of them significant, respectively, and 2 increases each, with 1 being significant for GBM. DT had 6 decreases and 1 significant increase, and LR had 4 decreases and 1 increase. Unlike the other classifiers, 3-NN had 5 increases in performance, however none of them were significant.

A summary of the results from Table 5.1 is shown on Table 5.2. Precision, Specificity, and Accuracy results consisted mainly of significant increases (83%, 81%, and 76%, respectively), with only 2 significant decreases and 6 non-significant decreases combined out of 126 cases. F_1 had 64% increases with majority of them significant and most decreases non-significant. AUROC had more decreases than increases, between significant and non-significant. On the other hand, Recall and G-Mean had a majority of decreases, 84% and 69% respectively. In total, there were 42% significant increases against 19% significant decreases, and the same percentage of non-significant increases and decreases. This highlights that *k*-INOS was beneficial to over-sampling algorithms most of the time (59%), considering the total number of measures, often in a statistically significant manner.

An additional examination of Table 5.1 reveals the number of performance metrics improved for each combination of classifier and over-sampling algorithm. The results are shown on Table 5.3. For two particular combinations (3-NN/ADASYN and DT/RWO), all 7 metrics improved with *k*-INOS. Similarly, in 6 out of 42 combinations, 6 out of 7 metrics improved and, in 8 out 42 cases, 5 metrics out of 7 improved. The most common number of metrics improved were 4, which happened in 15 out of 42 cases, and the remaining number of metrics improved (1, 2, and 3) accounted for 11 out of 42 cases. This indicates that *k*-INOS is usually beneficial, having increased the performance of at least 4 metrics in 31 out of 42 combinations of classifier and over-sampling algorithm.

Table 5.3: Performance metrics improved (out of 7) for each pair classifier/over-sampling algorithm.

	ROS	SMOTE	BDL-1	BDL-2	ADASYN	SL	RWO
LR	1	4	5	4	1	4	4
3-NN	1	5	6	6	**7**	6	6
DT	4	4	4	3	4	5	**7**
SVM	3	4	3	4	2	4	4
RF	4	3	3	4	4	2	5
GBM	5	5	5	5	6	6	1

The visualisations shown in Figures 5.8, 5.9, 5.10, and 5.11 help to get a better sense of the improvements provided by k-INOS. The 50 data sets are plotted in different colours depending on the change in performance of F_1, for a few combinations of classifier/over-sampling algorithm. Names in green represent performance increases, names in red represent performance decreases, and names in blue mean performance neither increased nor decreased. In Figure 5.8, the vast majority of data sets are green and performance gain was statistically significant. Figure 5.9 also has most data sets green, although performance increase was not significant. Figure 5.10 has an equal number of green and red data sets, however the red data sets are usually close to the boundary black line, which means that the performance degradation for these red data sets was small. On the other hand, several green data sets are far from the black line, which means that k-INOS largely improved performance. This is the case, for example, of yeast-0-2-5-6_vs_3-7-8-9. Finally, in Figure 5.11 most data sets are red and performance decreased significantly.

5.5 Summary

In this chapter we proposed k-INOS, a wrapper for over-sampling algorithms that enhances the robustness of the over-sampling algorithms to noisy examples in the minority class. k-INOS removes examples from the minority class likely to correspond to noise, prior to over-sampling, therefore providing a cleaner data set to the over-sampling algorithm. As minority examples are usually scarce and removed examples can be genuine, the removed examples are added back to the final over-sampled data set.

The experimentation conducted to assess the impact of k-INOS was extensive, including 50 imbalanced data sets, 6 classifiers, 7 over-sampling algorithms, and 7 performance metrics. k-INOS significantly increased the performance of the algorithms, most of the time, particularly for Accuracy, F_1, Precision, and Specificity. AUROC also experienced some performance improvements. On the other hand, Recall and G-Mean had mainly performance decreases. Moreover, it was observed that k-INOS worked better with 3-NN, DT, and GBM, although some improvements were also observed for LR, SVM, and RF.

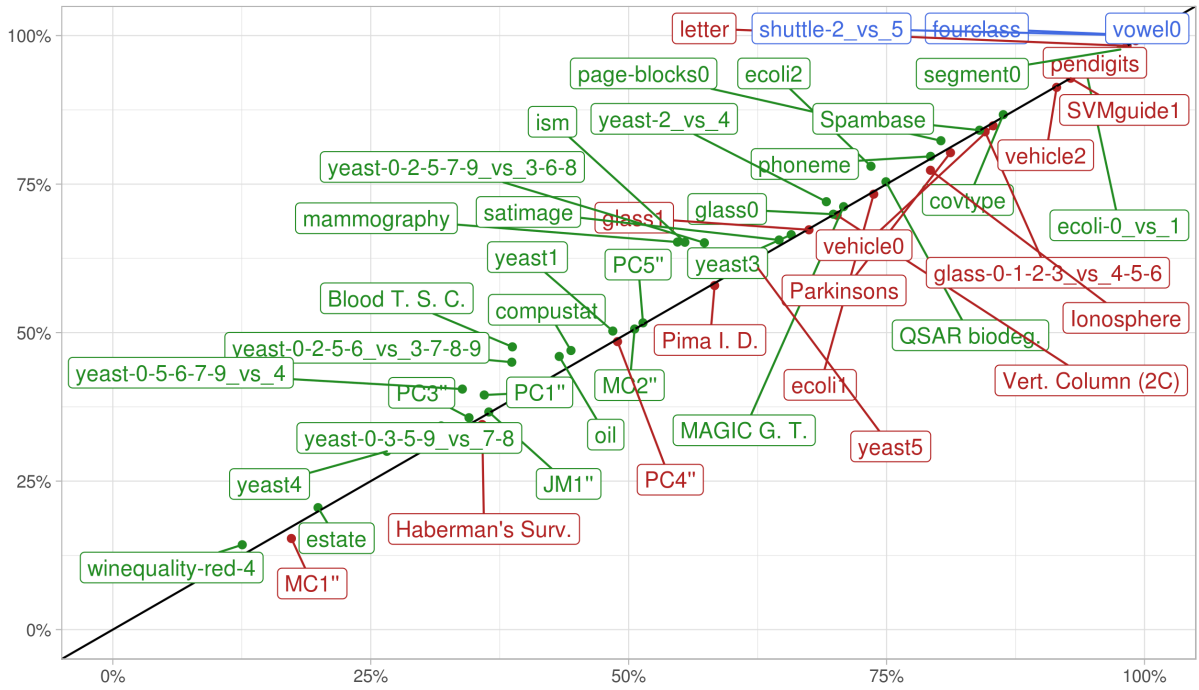

Figure 5.8: F_1 performance gains of SMOTE with k-INOS for the 50 data sets using 3-NN as base classifier.

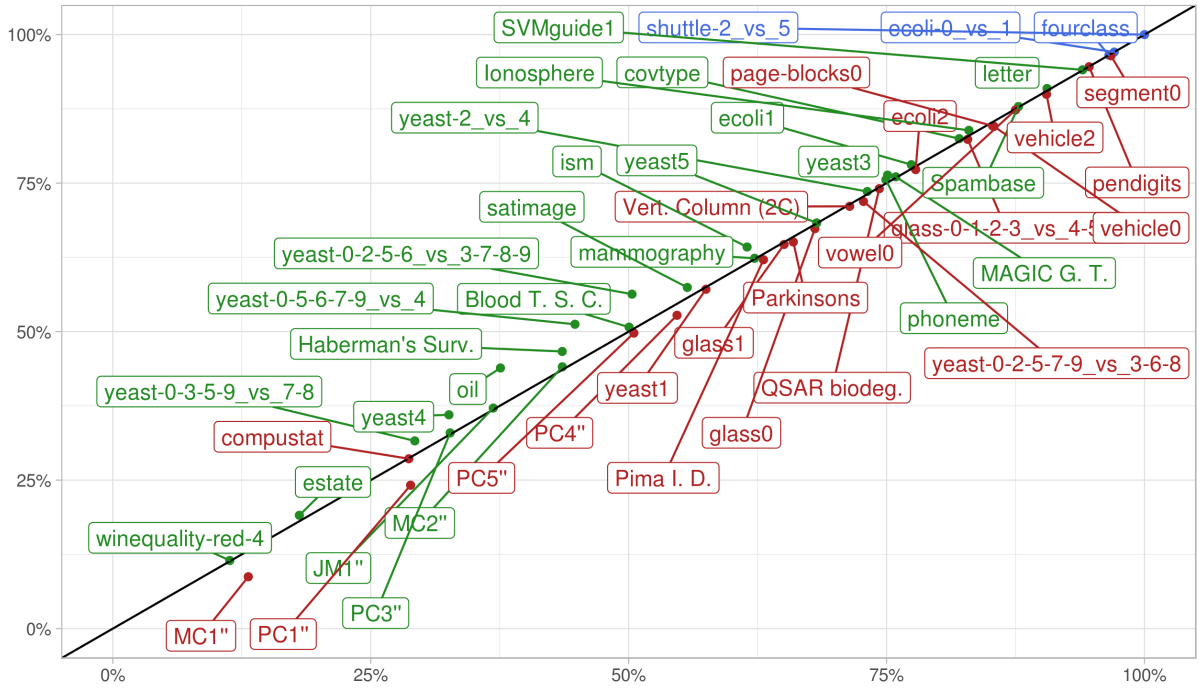

Figure 5.9: F_1 performance gains of SLSMOTE with k-INOS for the 50 data sets using DT as base classifier.

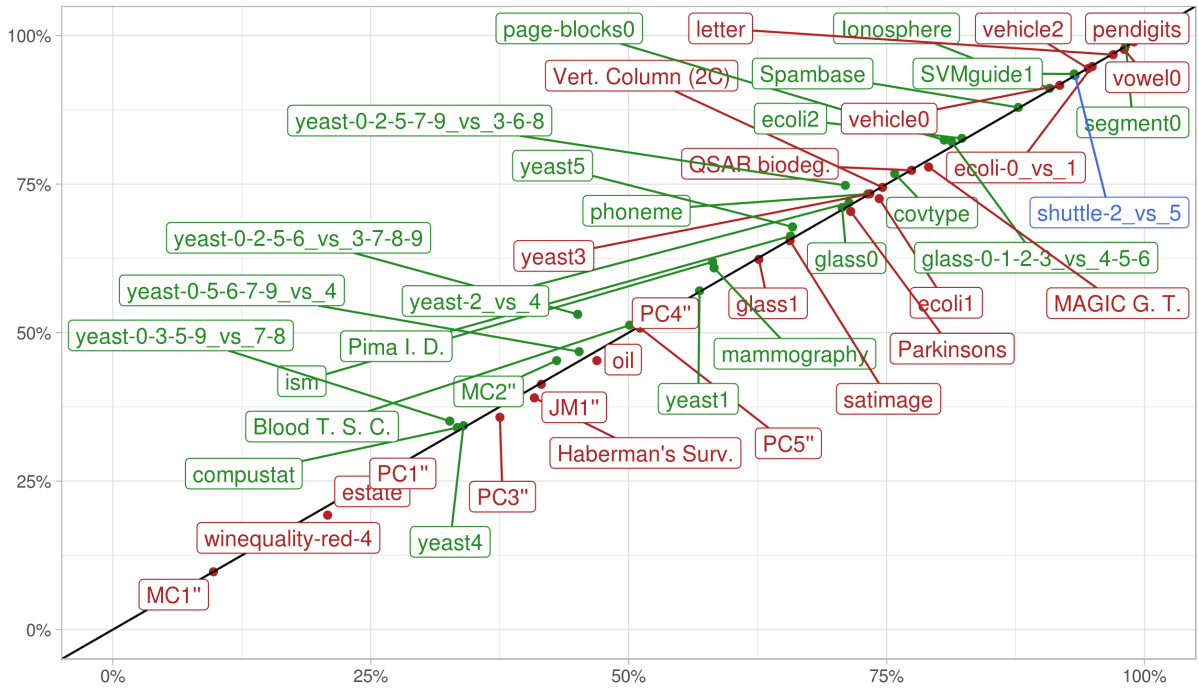

Figure 5.10: F_1 performance gains of BDLSMOTE-1 with k-INOS for the 50 data sets using SVM as base classifier.

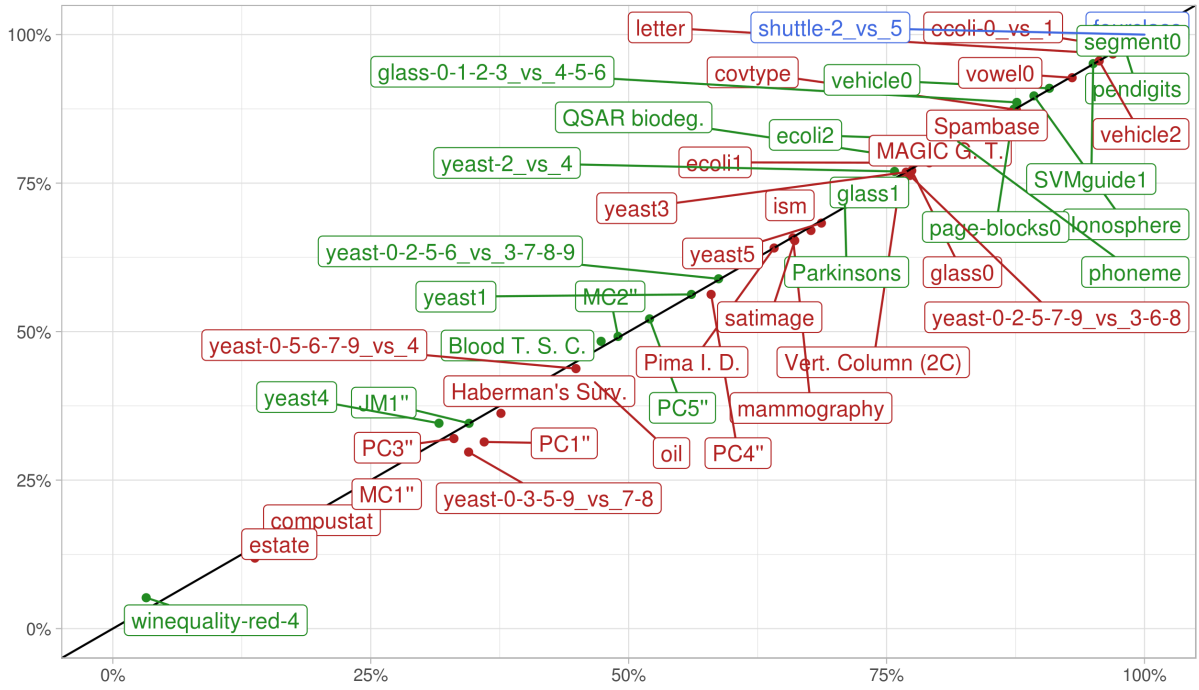

Figure 5.11: F_1 performance gains of ROS with k-INOS for the 50 data sets using RF as base classifier.

In the next chapter we explore *k*-INOS further by studying the importance of its hyperparameters and mining the results reported in this chapter to understand for which types of problems *k*-INOS is more likely to attain performance improvements.

6

k-INOS ANALYSIS

This chapter is devoted to increase our understanding of the *k*-INOS algorithm introduced in Chapter 5. To begin, we conduct a grid search over the hyperparameters of *k*-INOS using SMOTE as base over-sampling algorithm on the mammography data set. The aim of this grid search analysis is to better understand the importance of each hyperparameter of *k*-INOS and be able to suggest appropriate values for its use. Next, we mine the results of the experimentation of Chapter 5 to derive rules that capture which types of problems *k*-INOS is more likely to attain performance improvements, thus enabling us to know when it is appropriate to use *k*-INOS.

6.1 Analysis of k-INOS hyperparameters k and τ

To understand the importance of *k*-INOS' hyperparameters k and τ, a grid search was conducted on a range of values using SMOTE as base over-sampling algorithm, on the mammography data set (WOODS et al., 1993). The values $k = \{3,5,\ldots,15\}$ and $\tau = \{1,2,\ldots,5\}$ were explored. Note that $\tau = 1$ is the same as using SMOTE alone. Mean values of the performance metrics were computed using the same cross-validation scheme explained in Chapter 3. The results are shown in Figure 6.1.

An overall analysis of the results shows Accuracy, F_1, Precision, and Specificity benefiting from *k*-INOS in all configurations of k and τ. The opposite occurred with Recall and G-Mean, except for the LR classifier, where there were a few configurations for Recall and many configurations for G-Mean where *k*-INOS surpassed SMOTE alone. F_1, as the harmonic mean between Precision and Recall, is a relevant metric to observe their trade-off, and greatly improved with *k*-INOS for most classifiers. The only exception happened with RF, which attained smaller gains of F_1 when compared to the gains of other classifiers. AUROC generally increased for LR, 3-NN, and RF, and generally decreased for DT, GBM, and SVM, mostly with minimal changes. Even though observed results may vary with different data sets and over-sampling algorithms, they still give an indication on the appropriate values of k and τ to be used, depending on the performance metric to be optimised.

LR classifier: Accuracy increased with higher values of τ and slowly decreased with higher values of k, varying from a minimum of 89.55% (without *k*-INOS) to a maximum of

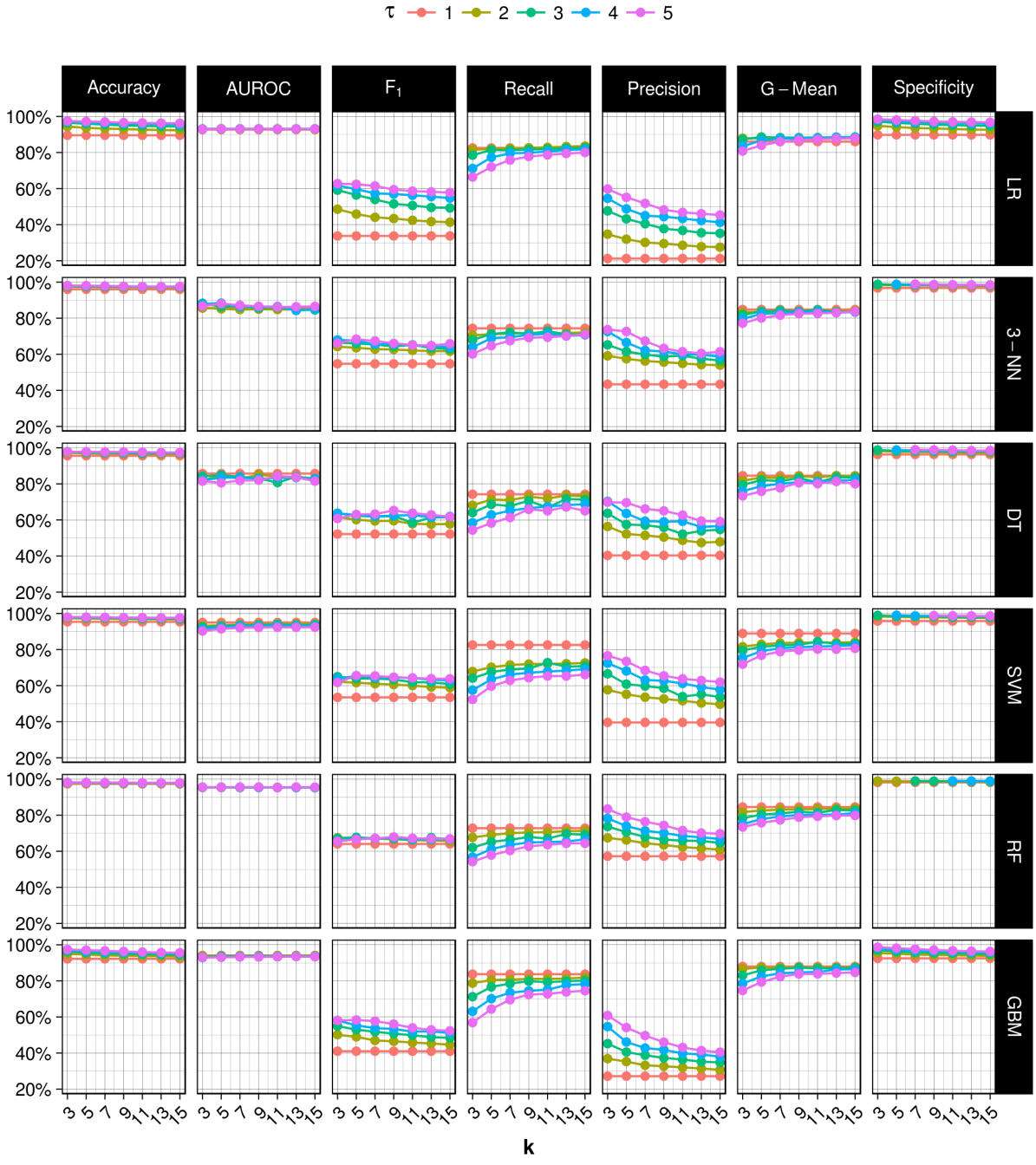

Figure 6.1: Grid search on k-INOS hyperparameters k and τ. For all classifiers, SMOTE was used as base over-sampling algorithm on the mammography data set.

97.47% ($k = 3$, $\tau = 5$). Precision greatly improved with higher values of τ and decreased with higher values of k, whereas Recall decreased with higher values of τ and increased with higher values of k. Precision ranged from 21.23% (without k-INOS) to 59.80% ($k = 3$, $\tau = 5$), while Recall ranged from 66.41% ($k = 3$, $\tau = 5$) to 83.56% ($k = 15$, $\tau = 2$). F_1 greatly increased with high values of τ and low values of k, varying from 33.76% (without k-INOS) to 62.76% ($k = 3$, $\tau = 5$). AUROC benefited from k-INOS in all setting but the gains were relatively small. Unlike the other classifiers, for LR, G-Mean increased for most configurations of k-INOS, increasing with higher values of k and increasing for $1 \leq \tau \leq 3$ and decreasing for $\tau > 3$. Finally, Specificity increased for higher values of τ and slowly decreased for higher values of k, varying from 89.78% (without k-INOS) to 98.50% ($k = 3$, $\tau = 5$).

3-NN classifier: Accuracy increased with higher values of τ and lower values of k, with variations ranging from 96.02% to 98.07%. Precision notably improved with higher values of τ and lower values of k, from 43.37% to 73.72%, while Recall fairly decreased with higher values of τ and lower values of k, from 74.39% to 60.16%. F_1 rose with higher values of τ and, in general, slightly decreased with higher values of k. F_1's maximum performance was 68.37% ($k = \tau = 5$) against a minimum of 54.73% (without k-INOS). Unlike the other classifiers, AUROC increased with high values of τ and low to moderate values of k, reaching a maximum of 88.35% ($k = 5$, $\tau = 4$), a minimum of 84.27% ($k = 13$, $\tau = 4$), and 85.57% with SMOTE alone. G-Mean increased with higher values of k and lower values of τ, achieving a maximum of 84.81% (without k-INOS) and a minimum of 77.22% ($k = 3$, $\tau = 5$). Specificity increased with higher values of τ and slowly decreased with higher values of k, achieving a maximum of 99.28% ($k = 3$, $\tau = 5$) and a minimum of 96.74% (without k-INOS).

DT classifier: Increasing the value of τ led to higher Accuracy, but for $\tau \geq 2$ the gains were smaller. On the other hand, increasing values of τ led to considerable gains of Precision and moderate losses of Recall. Precision varied from 40.33% (SMOTE alone) to 70.32% ($k = 3$, $\tau = 4$) and Recall varied from 74.23% (SMOTE alone) to 54.41% ($k = 3$, $\tau = 5$). Increasing k had the opposite effect of τ on Precision and Recall, where higher values of k increased Recall and decreased Precision. The trade-off between Precision and Recall, captured through F_1, benefited from high values of τ and low to moderate values of k. Minimum F_1 performance was 52.17% (SMOTE alone) and maximum performance was 65.24% ($k = 9$, $\tau = 5$). AUROC steadily decreased from 85.01% to 80.58% for $\tau \geq 2$, without any consistent pattern of variation with respect to hyperparameter k. G-Mean increased with higher values of k and lower values of τ, ranging from 73.35% ($k = 3$, $\tau = 5$) to 84.57% ($k = 13$, $\tau = 2$), and Specificity improved with higher values of τ and slowly decreased with higher values of k, varying from 96.30% (without k-INOS) to 99.20% ($k = 3$, $\tau = 5$).

SVM classifier: Accuracy increased with higher values of τ and lower values of k. Precision greatly increased with higher values of τ and lower values of k, from 39.60% to 76.59%, while Recall decreased, from 82.53% to 52.35%. F_1 increased with higher values of τ, from a minimum of 53.48% to a maximum of 65.58% ($k = \tau = 5$). AUROC degraded for higher

values of τ and lower values of k, varying from 90.26% to 95%. G-Mean steadily increased with higher values of k and decreased with higher values of τ, while the opposite happened to the Specificity. G-Mean values varied from 80.81% ($k = 3$, $\tau = 5$) to 88.61 (without k-INOS), and Specificity values varied from 89.78% (without k-INOS) to 98.50 ($k = 3$, $\tau = 5$).

RF classifier: Accuracy increased, particularly for higher values of τ and lower values of k, with marginal gains in performance, though. Higher values of τ and lower values of k led to higher gains in Precision while the opposite happened to Recall. Precision ranged from 57.26% to 83.44% and Recall ranged from 72.81% to 54.24%. F_1 also increased in all settings, although with smaller gains when compared to other classifiers, ranging from 64.06% to 67.94%. AUROC mainly increased for most settings, however with negligible gain variations. G-Mean steadily increased with higher values of k and lower values of τ, ranging from 73.43% ($k = 3$, $\tau = 5$) to 84.53% (without k-INOS). Specificity benefited from low values of k and high values of τ, but the performance gains were negligible.

GBM classifier: There was a clear trend in the results. Accuracy benefited from low values of k and high values of τ and varied from 92.22% to 97.38%. Precision greatly improved with higher values of τ, from 27.18% to 60.81%, while Recall deteriorated, from 83.72% to 56.85%. k had the opposite effect of τ on Precision and Recall, although with less impact for small values of τ. F_1 increased with higher values of τ and generally decreased with higher values of k. Maximum performance for F_1 was 58.34% ($k = \tau = 5$) and minimum was 41.02% (without k-INOS). AUROC did not vary considerably, ranging from 92.9% to 94.10%, where the maximum was achieved with SMOTE alone. G-Mean increased with higher values of k and lower values of τ, ranging from 74.67% ($k = 3$, $\tau = 5$) to 87.99% (without k-INOS), and Specificity increased with higher values of τ and lower values of k, ranging from 92.50% (without k-INOS) to 98.73% ($k = 3$, $\tau = 5$).

6.2 Mining *k*-INOS Performance

Since the results of the over-sampling algorithms, in Chapter 5, varied according to different metrics, rules were extracted to mine the likely conditions for performance changes. The extraction of rules was performed for each performance metric by converting the results into sets of cases. Each case is composed of a data set, a classifier, an over-sampling algorithm, and a flag indicating whether k-INOS improved performance or not. Furthermore, each data set is characterised by a set of 8 features and 9 complexity measures, described in Table 6.1. The features consist of potentially useful information for representing conditions and the complexity measures have been employed in the literature to describe data set characteristics (MICHIE; SPIEGELHALTER; TAYLOR, 1994; HO; BASU, 2002). The idea is to identify if, prior to application of k-INOS, any particular conditions from the data sets can point out towards performance change. A decision tree based on CART (BREIMAN et al., 1984) was induced on each set of cases to extract rules for analysis.

Table 6.1: Features and complexity measures employed to describe the data sets.

Name	Description
num_examples	Number of examples.
num_examples_majority	Number of examples in the majority class.
num_examples_minority	Number of examples in the minority class.
num_features	Number of features.
num_features_binary	Number of binary features.
proportion_examples_majority	Proportion of examples in the majority class.
proportion_examples_minority	Proportion of examples in the minority class.
proportion_features_binary	Proportion of binary features.
corr_abs	Mean absolute correlation coefficient. Reflects the level of information in the features.
sd_ratio	Geometric mean ratio of standard deviations. Reflects the level of information in the features.
F1	Fisher's discriminant ratio. Reflects the level of information in the features.
F2	Volume of overlap region. Reflects class overlap.
N2	Ratio of average intra/inter class 1-NN distance. Reflects class overlap.
N3	Leave-one-out cross-validation (loocv) error rate of 1-NN classifier. Reflects complexity.
N4	Nonlinearity of the 1-NN classifier. Reflects complexity.
T2	Average number of points per dimension. Reflects density of feature space.
IR	Imbalance ratio.

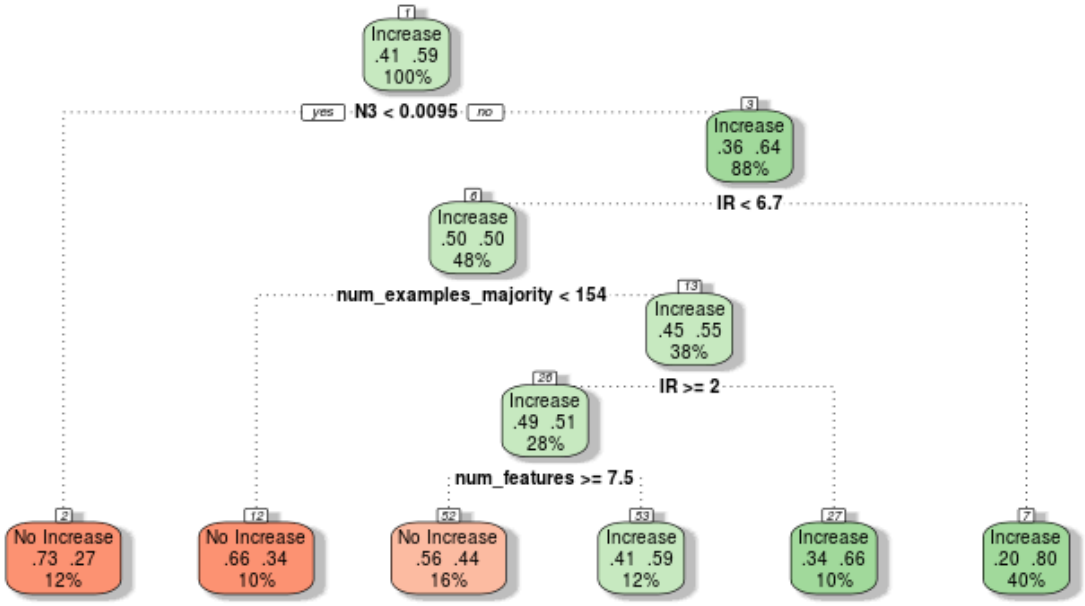

Figure 6.2: Decision tree induced from the set of cases for Accuracy.

Figures 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, and 6.8 show the trees induced, according to each performance metric. To avoid creating very specific rules, each leaf node required to have at least 10% of all cases, except for the Recall metric, which failed to create a tree following this condition and was allowed to build a tree with leaf nodes containing at least 5% of all cases. Each node displays the feature analysed, the proportion of situations (improvements or no improvements) in each side and the percentage of cases evaluated at that point. The greener the nodes the higher the proportion of improvements, whereas the redder the nodes the higher the proportion of no improvements.

Overall, the classification complexity of a data set, as measure by the loocv error rate of the 1-NN classifier (N3), seemed to be the most informative descriptor to judge whether *k*-INOS would improve performance or not. In addition, the level of information present in the features (F1), the imbalance ratio (IR), and the density of feature space (T2) were also common predictors of performance improvement. It is interesting to note as well that several data set descriptors did not appear in any induced tree. These were: num_examples, num_examples_minority, num_features_binary, proportion_examples_majority, proportion_features_binary, sd_ratio, F2, and N2.

The tree for Accuracy is shown in Figure 6.2. A performance improvement happens in 59% of the cases. The most expressive rule for improvement of Accuracy, which comprises 40% of the cases, can be interpreted as: data sets that are not trivial to classify (N3 $\geq$ 0.0095) and are highly imbalanced (IR $\geq$ 6.70) are likely to benefit from *k*-INOS 80% of the time. On the other hand, the most expressive rule for no improvement of Accuracy, which comprises 12% of the cases, can be interpreted as: data sets that are trivial to classify (N3 < 0.0095) are unlikely to

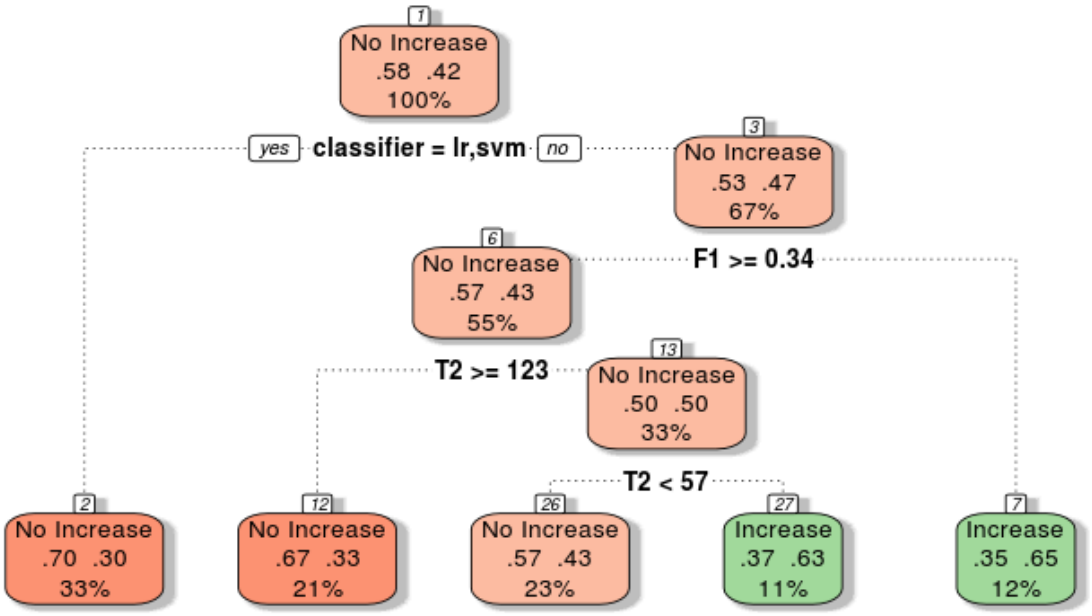

Figure 6.3: Decision tree induced from the set of cases for AUROC.

benefit from *k*-INOS 73% of the time.

The tree for AUROC is shown in Figure 6.3. For 58% of the cases, AUROC did not improve by using *k*-INOS. In two out of the five rules derived by the tree, AUROC increased for majority of cases. These two rules comprise 23% of the cases and the rightmost one can be interpreted as: data sets that were not used to train a LR or SVM and do not have any highly informative feature (F1 < 0.34) are likely to benefit from *k*-INOS 65% of the time. The most expressive rule for no improvement of AUROC comprises 33% of the cases and states that if a data set was used to train a LR or SVM, it is unlikely to benefit from *k*-INOS 70% of the time.

The tree for Precision is shown in Figure 6.4. Overall, 62% of the cases improved Precision by using *k*-INOS. Two out of the four rules derived by the tree had mostly improvements and comprised 73% of the cases, whereas the other two rules had mostly no improvements and comprised 27% of the cases. The most expressive rule for improvement of Precision, which comprises 36% of the cases, can be interpreted as: data sets that are not trivial to classify (N3 $\geq$ 0.0095) and are highly imbalanced (IR $\geq$ 7.7) are likely to benefit from *k*-INOS 82% of the time. The most expressive rule for no improvement of Precision, which comprises 12% of the cases, states that data sets that are trivial to classify (N3 < 0.0095) are unlikely to benefit from *k*-INOS 66% of the time.

The tree for Recall is shown in Figure 6.5. Majority of the derived rules comprise mostly no improvements for Recall. The only rule that had more improvements than no improvements comprised 7% of all cases and can be interpreted as: data sets that are not over-sampled by ROS or SMOTE, not trivial to classify (N3 $\geq$ 0.18), and have a dense feature space (T2 $\geq$ 107), are likely to benefit from *k*-INOS 62% of the time. The most expressive rule for no improvement,

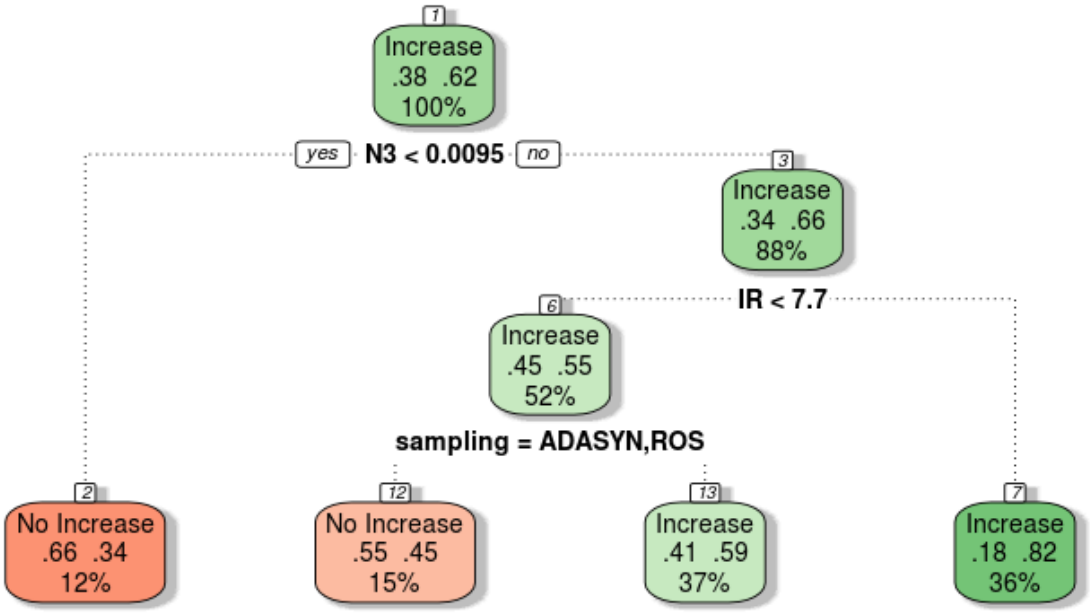

Figure 6.4: Decision tree induced from the set of cases for Precision.

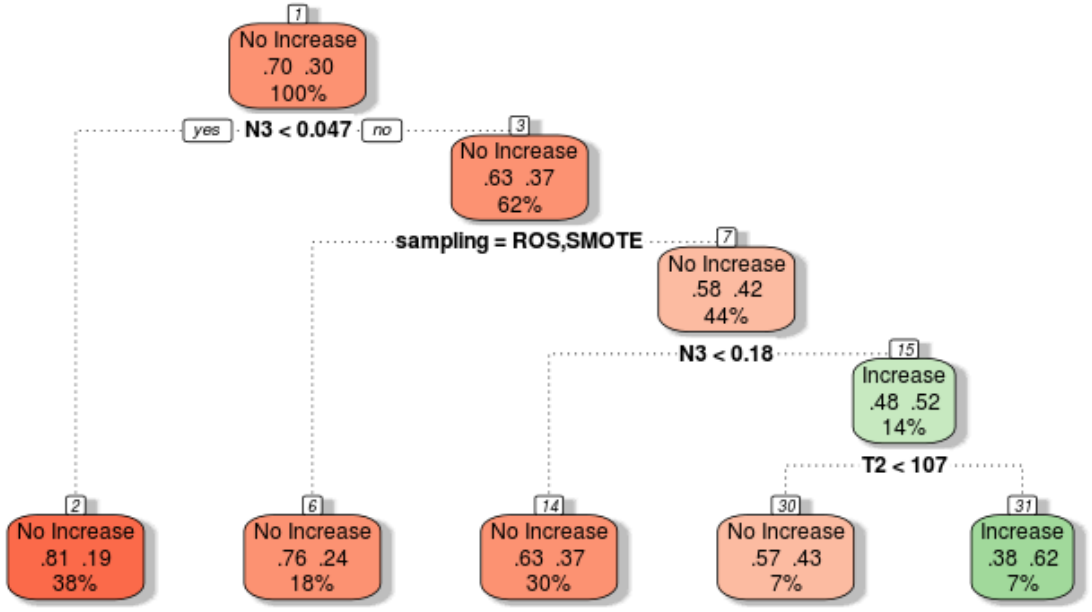

Figure 6.5: Decision tree induced from the set of cases for Recall.

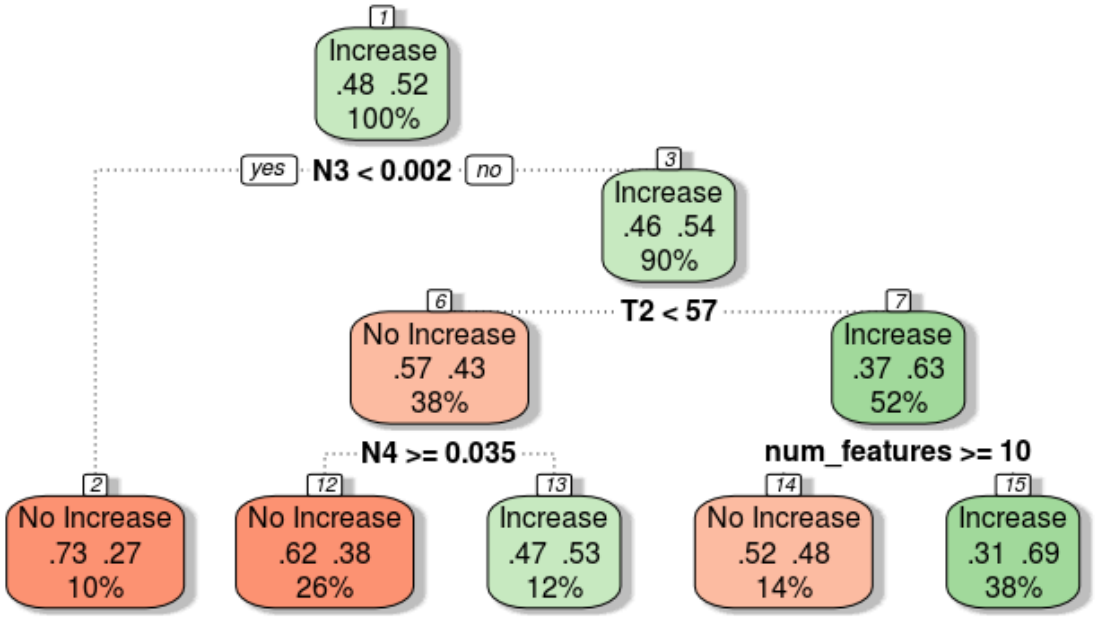

Figure 6.6: Decision tree induced from the set of cases for F_1.

which comprises 38% of all cases, states that data sets that are trivial to classify (N3 < 0.047) are unlikely to benefit from *k*-INOS 81% of the time.

The tree for F_1 is shown in Figure 6.6. Over half the cases, 52% of them, had an improvement in performance by using *k*-INOS. Two out of the five rules derived by the tree mostly accounted for improvements and comprised 50% of the cases. The most expressive rule for improvement, comprising 38% of the cases, can be interpreted as: data sets that are not trivial to classify (N3 $\geq$ 0.002), have a dense feature space (T2 $\geq$ 57), and are low dimensional (number of features < 10), benefit from *k*-INOS 69% of the time. The other three rules mostly had no improvements and accounted for the other 50% of the cases. One of them, which comprises 10% of the cases, states that data sets that are trivial to classify (N3 < 0.002) are unlikely to benefit from *k*-INOS 73% of the time.

The tree for G-Mean is shown in Figure 6.7. In general, G-Mean did not benefit from *k*-INOS in 62% of the cases. Out of the four rules derived, only one was composed mostly of improvements. This single rule for improvement of G-Mean, which comprised 18% of the cases, can be interpreted as: data sets without a high imbalance (IR < 10), that have a fair number of independent features (corr_abs < 0.35), and do not have any highly informative feature (F1 < 0.85), benefited from *k*-INOS 62% of the time. Of the three rules for no improvement, the most expressive one comprised 24% of the cases and states that highly imbalanced data sets are unlikely to benefit from *k*-INOS 80% of the time.

The tree for Specificity is shown in Figure 6.8. Specificity mostly benefited from *k*-INOS where 64% of the cases had an improvement in performance. Two out of the five rules derived by the tree had mostly improvements and accounted for 66% of the cases. The most expressive

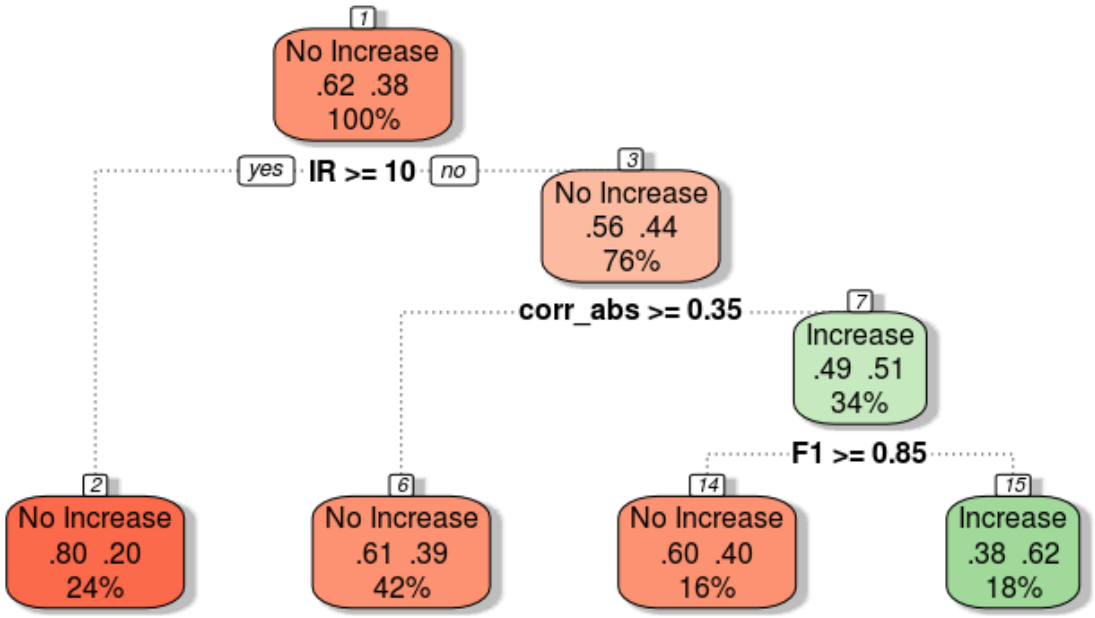

Figure 6.7: Decision tree induced from the set of cases for G-Mean.

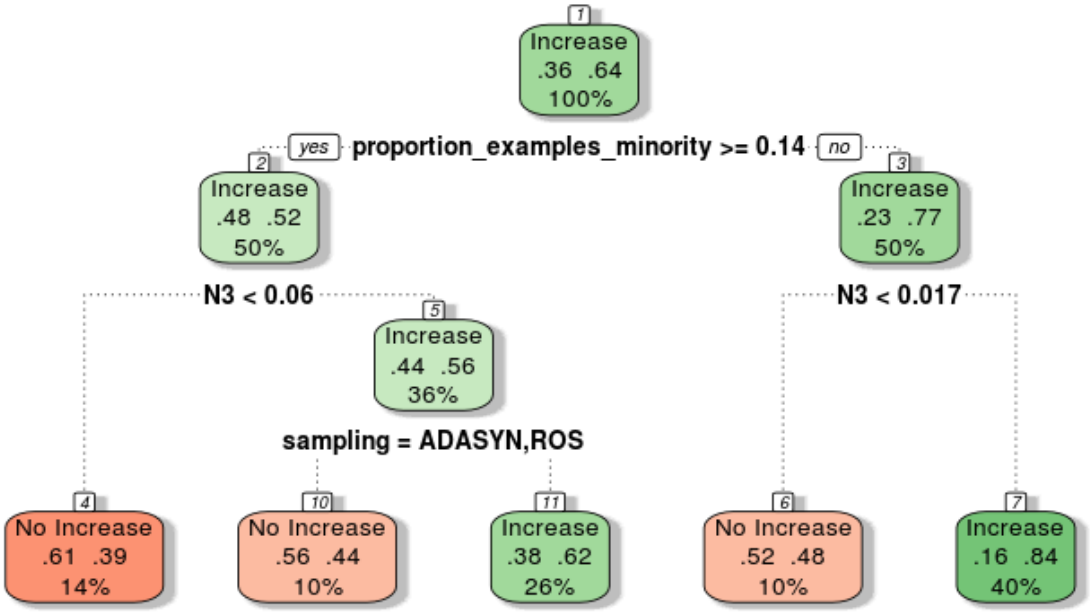

Figure 6.8: Decision tree induced from the set of cases for Specificity.

rule for improvement, which accounted for 40% of the cases, can be interpreted as: data sets that have a low proportion of minority examples (proportion_minority_examples < 0.14) and are not trivial to classify (N3 $\geq$ 0.017), are likely to benefit from k-INOS 84% of the time. On the other hand, the most expressive rule for no improvement that accounted for 14% of the cases, can be interpreted as: data sets that do not have a low proportion of minority examples (proportion_minority_examples $\geq$ 0.14) and are trivial to classify (N3 < 0.06), are unlikely to benefit from k-INOS 61% of the time.

6.3 Summary

In this chapter we explored further the k-INOS algorithm by analysing its hyperparameters and mining the results obtained by k-INOS in Chapter 5.

To understand the importance of k-INOS' hyperparameters k and τ, an independent study was conducted on the mammography data set. A grid search on a range of values for k and τ was performed using SMOTE as base over-sampling algorithm. For all classifiers, Accuracy, Precision, F_1, and Specificity increased for all settings explored, highlighting the benefit of using k-INOS. Low to moderate values of k (5 to 11) and moderate values of τ (3 or 4) seemed to work well in most cases of this study.

In addition, a set of rules were mined from the results of Chapter 5 to provide insights on which types of data sets and conditions k-INOS is more likely to attain performance improvements. For most performance metrics, the loocv error rate of the 1-NN classifier (N3) was the most important complexity metric to determine performance improvement. The imbalance ratio (IR), the Fisher's discriminant ratio (F1), and the density of examples in feature space (T2) were also common predictors of k-INOS impact. Moreover, several complexity metrics such as the geometric mean ratio of standard deviations (sd_ratio) and the volume of overlap region (F2) did not appear in any of the trees mined.

7

CONCLUSIONS

In this dissertation we investigated the problem of learning from imbalanced data sets and proposed two new *data sampling* algorithms: RRUS and k-INOS. Both algorithms were extensively tested on 50 imbalanced data sets, 6 classifiers, and evaluated on 7 performance metrics. Mean values of the performance metrics were computed using a 5×2-fold cross-validation procedure.

RRUS is an under-sampling algorithm and is roughly based on the RUS and KMUS algorithms. RRUS' objective is to select a subset of examples from the majority class that appropriately represents the original majority class. By an appropriate representation we mean that the subset of examples has an estimated density distribution similar to the density distribution of the majority class. To achieve that, RRUS samples many random subsets of examples from the majority class and picks the subset that has a centroid closer to the centroid of the majority class. As density distributions can be multimodal, RRUS first clusters the majority class with the k-Means algorithm and applies the aforementioned procedure to each cluster. A comprehensive experimentation was conducted and compared RRUS to other 3 under-sampling algorithms: RUS, KMUS, and SBC. The statistical significance of the results was assessed by the *Friedman Aligned Ranks* test and the *Finner* post hoc test. For most classifiers and performance metrics RRUS achieved the best average rank, and it was significantly better than KMUS and SBC most of the time, and significantly better than RUS many times. Only for the Recall metric RRUS did not perform well, however, RRUS achieved the best average ranks for the F_1 metric for 5 out of 6 classifiers, which shows that it greatly improved the Precision/Recall trade-off most of the time. Unlike RUS that works for any type of data set, RRUS only works with data sets containing numerical attributes, as RRUS needs to compute centroids of subsets of examples.

k-INOS is a general strategy to enhance robustness of over-sampling algorithms to noise. It removes noisy minority examples present in the data prior to over-sampling and returns them to the data after over-sampling. The rationale behind k-INOS is that noisy examples may severely affect over-sampling algorithms as they might use these noisy examples to generate new examples. However, imbalanced data sets usually have few examples in the minority class to start with, therefore, the removed examples are added back to the data set after over-sampling. In addition, noise is an ill-defined concept and removed examples could potentially be genuine

examples. To assess the efficacy of k-INOS we tested it on 7 over-sampling algorithms and evaluated if the difference in performance was statistically significant using a Wilcoxon signed-ranks test. For most combinations of classifier and over-sampling algorithm, k-INOS significantly improved performance of Accuracy, Precision, and Specificity. Moreover, the F_1 metric also experienced many improvements in performance, where over 35% of them were significant. The behaviour of k-INOS is controlled by two hyperparameters: k and τ. To analyse which values of k and τ should be used in practice, a grid search on a range of values of k and τ was conducted using SMOTE as base over-sampling algorithm on the mammography data set. The results showed that low to moderate values of k (5 to 11) and moderate values of τ (3 or 4) work well in practice. Finally, to understand in which situations k-INOS is more likely to attain performance improvements a set of rules were extracted for each performance metric, from the main experimentation's results, and in general they showed that the N3 complexity metric (loocv error rate of the 1-NN classifier) is a common indicator of whether k-INOS is likely to attain performance improvements or not.

7.1 Limitations

Although we tried to be as comprehensive as possible with the work developed in this dissertation, several limitations were present.

First, we must note that all experiments were conducted on 50 imbalanced data sets chosen according to a few criteria. We expect that the results obtained extrapolate to other imbalanced data sets, however we can only claim them to do on imbalanced data sets with characteristics similar to the selected ones. Second, for each classifier employed we fixed its hyperparameters' values, as we were interested in the performance improvements attained by the sampling algorithms and not by the classifiers. However, a highly tuned classifier may benefit more or less from the proposed sampling algorithms, but unfortunately this issue is not addressed here. Third, we tried to evaluate the results from as many different angles as possible by selecting 7 performance metrics that we believe to cover several angles well. Therefore, even though more performance metrics could lead to a finer understanding of the results, we believe the increased refinement would not be worth the extra analysis.

From the point of view of the proposed methods, to assert that they are state-of-the-art they would have to be compared to a wider range of sampling algorithms. In the case of RRUS, we only compared it to other 3 under-sampling algorithms because they were the most similar to RRUS and the *Friedman Aligned Ranks* test is recommended when comparing at most 4 or 5 algorithms (GARCÍA et al., 2010). k-INOS, on the other hand, had its efficacy demonstrated by testing it on 7 over-sampling algorithms, however, no comparisons were made between k-INOS and other noisy removal methods. This is left as a future work.

7.2 Future Works

For both proposals, RRUS and k-INOS, the Recall metric did not seem to benefit from them. In the case of RRUS, it frequently achieved the worst mean rank when compared to the other under-sampling algorithms. For k-INOS, most over-sampling algorithms had a significant decrease in Recall when combined with k-INOS. Note, however, that any classifier can achieve maximum Recall performance by predicting every example to belong to the minority class, which is clearly not a desirable strategy. Therefore, an investigation into the low performance of both proposals on the Recall metric is an interesting possibility for future work, but Recall should be always analysed in conjunction with Precision and F_1 metrics to avoid situations as the one just described.

The classifiers selected for the experiments were picked in a way to be as diverse as possible. Classifiers representing different paradigms were chosen, however, there were still some learning paradigms that did not have a classifier to represent them in the experiments. For instance, a probabilistic classifier such as the Naive Bayes. These extra classifiers are left as future work to be included in the experiments and analysed.

In regards to RRUS, a first direction for future work is to adapt the idea of RRUS to other stochastic under-sampling algorithms. For instance, by applying the selection strategy of these other under-sampling algorithms to sample the subsets of examples from the majority class. A second direction is to employ discrete optimisation to select a subset of examples that has a centroid closer to the majority class' centroid. For example, instead of randomly searching through $\binom{m^-}{m}$ subsets of examples, a genetic algorithm could be used for that purpose. A third direction is an analysis of the influence of the clusters utilised by RRUS in its performance. Would a different clustering algorithm lead to different results? Another direction is a study of RRUS's hyperparameters, in special the relationship between the sample size m and the number of randomly selected subsets $times$. The discrete function $f(m) = \binom{m^-}{m}$, for $0 < m < m^-$, is monotonically increasing for $0 < m \leq \left\lfloor \frac{m^-}{2} \right\rfloor$ and monotonically decreasing for $\left\lceil \frac{m^-}{2} \right\rceil \leq m < m^-$, and achieves its maximum at $m = \left\lceil \frac{m^-}{2} \right\rceil$ (or $m = \left\lfloor \frac{m^-}{2} \right\rfloor$ due to symmetry) and its minimum at $m = 1$ (or $m = m^- - 1$ due to symmetry as well), therefore it is reasonable to expect higher values of $times$ for values of m around $\left\lceil \frac{m^-}{2} \right\rceil$ for RRUS to work well.

For k-INOS there are several ideas for further exploration. First, the concept of neighbourhood of influence could be used to create new over-sampling and under-sampling algorithms. Second, the distribution of the sizes of the modified neighbourhood of influence could be studied to give insights into the nature of the imbalanced data sets. For instance, it was informally noticed that data sets with many minority examples with small modified neighbourhoods of influence were harder to achieve high values of F_1 measure.

7.3 Publication Disclosure

Parts of this dissertation were already published or submitted for consideration in upcoming conferences and journals. Here, we list these articles and the associated parts of this dissertation.

- Published:

  (Chapter 5) **de Morais**, **R. F.**, & Vasconcelos, G. C. (2017, August). **Under-Sampling the Minority Class to Improve the Performance of Over-Sampling Algorithms in Imbalanced Data Sets**. In *Workshop on Learning in the Presence of Class Imbalance and Concept Drift (LPCICD'17)*.

- Submitted:

  (Chapter 4) **de Morais**, **R. F.**, & Vasconcelos, G. C.. **Improving the Performance of Classifiers on Imbalanced Data Sets by Preserving the Density Distribution of the Majority Class**. Submitted for consideration at *International Joint Conference on Artificial Intelligence (IJCAI'18)*.

  (Chapters 5 and 6) **de Morais**, **R. F.**, & Vasconcelos, G. C.. **Boosting the Performance of Over-Sampling Algorithms through Under-Sampling the Minority Class**. Submitted for consideration at *Neurocomputing: Special Issue on Learning in the Presence of Class Imbalance and Concept Drift*.

REFERENCES

ABU-MOSTAFA, Y. S.; MAGDON-ISMAIL, M.; LIN, H.-T. **Learning from data**. [S.l.]: AMLBook New York, NY, USA:, 2012. v.4.

ALCALÁ, J. et al. Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. **Journal of Multiple-Valued Logic and Soft Computing**, [S.l.], v.17, n.2-3, p.255–287, 2010.

ALI, K.; PAZZANI, M. Reducing the Small Disjuncts Problem by Learning Probabilistic Concept Descriptions, Computational Learning Theory and Natural Learning Systems, T. **Petsche et al.(Eds.)**, [S.l.], v.3, 1992.

ALIBEIGI, M.; HASHEMI, S.; HAMZEH, A. DBFS: an effective density based feature selection scheme for small sample size and high dimensional imbalanced data sets. **Data & Knowledge Engineering**, [S.l.], v.81, p.67–103, 2012.

BATISTA, G. E.; PRATI, R. C.; MONARD, M. C. A study of the behavior of several methods for balancing machine learning training data. **ACM Sigkdd Explorations Newsletter**, [S.l.], v.6, n.1, p.20–29, 2004.

BELLINGER, C. **Beyond the boundaries of smote**: a framework for manifold-based synthetic oversampling. 2016. Tese (Doutorado em Ciência da Computação) — Université d'Ottawa/University of Ottawa.

BREIMAN, L. Random forests. **Machine learning**, [S.l.], v.45, n.1, p.5–32, 2001.

BREIMAN, L. et al. **Classification and regression trees**. [S.l.]: CRC press, 1984.

BUNKHUMPORNPAT, C.; SINAPIROMSARAN, K. DBMUTE: density-based majority under-sampling technique. **Knowledge and Information Systems**, [S.l.], v.50, n.3, p.827–850, 2017.

BUNKHUMPORNPAT, C.; SINAPIROMSARAN, K.; LURSINSAP, C. Safe-level-smote: safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In: PACIFIC-ASIA CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING. **Anais...** [S.l.: s.n.], 2009. p.475–482.

BUNKHUMPORNPAT, C.; SINAPIROMSARAN, K.; LURSINSAP, C. DBSMOTE: density-based synthetic minority over-sampling technique. **Applied Intelligence**, [S.l.], v.36, n.3, p.664–684, 2012.

BUREZ, J.; POEL, D. Van den. Handling class imbalance in customer churn prediction. **Expert Systems with Applications**, [S.l.], v.36, n.3, p.4626–4636, 2009.

CARVALHO, D. R.; FREITAS, A. A. A hybrid decision tree/genetic algorithm for coping with the problem of small disjuncts in data mining. In: ANNUAL CONFERENCE ON GENETIC AND EVOLUTIONARY COMPUTATION, 2. **Proceedings...** [S.l.: s.n.], 2000. p.1061–1068.

CHAN, P. K. et al. Distributed data mining in credit card fraud detection. **IEEE Intelligent Systems and Their Applications**, [S.l.], v.14, n.6, p.67–74, 1999.

CHAWLA, N. V. et al. SMOTE: synthetic minority over-sampling technique. **Journal of artificial intelligence research**, [S.l.], v.16, p.321–357, 2002.

CHEN, T. et al. **xgboost**: extreme gradient boosting. [S.l.: s.n.], 2017. R package version 0.6.4.6.

CIESLAK, D. A.; CHAWLA, N. V.; STRIEGEL, A. Combating imbalance in network intrusion datasets. In: GRC. **Anais...** [S.l.: s.n.], 2006. p.732–737.

CIESLAK, D. A. et al. Hellinger distance decision trees are robust and skew-insensitive. **Data Mining and Knowledge Discovery**, [S.l.], v.24, n.1, p.136–158, 2012.

COVER, T.; HART, P. Nearest neighbor pattern classification. **IEEE transactions on information theory**, [S.l.], v.13, n.1, p.21–27, 1967.

DAL POZZOLO, A. et al. Learned lessons in credit card fraud detection from a practitioner perspective. **Expert systems with applications**, [S.l.], v.41, n.10, p.4915–4928, 2014.

DAS, B.; KRISHNAN, N. C.; COOK, D. J. RACOG and wRACOG: two probabilistic oversampling techniques. **IEEE transactions on knowledge and data engineering**, [S.l.], v.27, n.1, p.222–234, 2015.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. **Journal of Machine learning research**, [S.l.], v.7, n.Jan, p.1–30, 2006.

DENIL, M.; TRAPPENBERG, T. P. Overlap versus imbalance. In: CANADIAN CONFERENCE ON AI. **Anais...** [S.l.: s.n.], 2010. p.220–231.

FAWCETT, T. An Introduction to ROC Analysis. **Pattern Recogn. Lett.**, New York, NY, USA, v.27, n.8, p.861–874, June 2006.

FINNER, H. On a monotonicity problem in step-down multiple test procedures. **Journal of the American Statistical Association**, [S.l.], v.88, n.423, p.920–923, 1993.

FRIEDMAN, J. H. Greedy function approximation: a gradient boosting machine. **Annals of statistics**, [S.l.], p.1189–1232, 2001.

GALAR, M. et al. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. **IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)**, [S.l.], v.42, n.4, p.463–484, 2012.

GAO, M. et al. PDFOS: pdf estimation based over-sampling for imbalanced two-class problems. **Neurocomputing**, [S.l.], v.138, p.248–259, 2014.

GARCÍA, S. et al. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. **Information Sciences**, [S.l.], v.180, n.10, p.2044–2064, 2010.

GU, Q.; ZHU, L.; CAI, Z. Evaluation measures of the classification performance of imbalanced data sets. **Computational intelligence and intelligent systems**, [S.l.], p.461–471, 2009.

HA, J.; LEE, J.-S. A New Under-Sampling Method Using Genetic Algorithm for Imbalanced Data Classification. In: INTERNATIONAL CONFERENCE ON UBIQUITOUS INFORMATION MANAGEMENT AND COMMUNICATION, 10. **Proceedings...** [S.l.: s.n.], 2016. p.95.

HAIXIANG, G. et al. Learning from class-imbalanced data: review of methods and applications. **Expert Systems with Applications**, [S.l.], 2016.

HAN, H.; WANG, W.-Y.; MAO, B.-H. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: INTERNATIONAL CONFERENCE ON INTELLIGENT COMPUTING. **Anais...** [S.l.: s.n.], 2005. p.878–887.

HART, P. The Condensed Nearest Neighbor Rule (Corresp.). **IEEE Trans. Inf. Theor.**, Piscataway, NJ, USA, v.14, n.3, p.515–516, Sept. 2006.

HE, H. et al. 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. **Anais...** [S.l.: s.n.], 2008. p.1322–1328.

HE, H.; GARCIA, E. A. Learning from imbalanced data. **IEEE Transactions on knowledge and data engineering**, [S.l.], v.21, n.9, p.1263–1284, 2009.

HELLEPUTTE, T. **LiblineaR**: linear predictive models based on the liblinear c/c++ library. [S.l.: s.n.], 2017. R package version 2.10-8.

HO, T. K.; BASU, M. Complexity measures of supervised classification problems. **IEEE transactions on pattern analysis and machine intelligence**, [S.l.], v.24, n.3, p.289–300, 2002.

HODGES, J.; LEHMANN, E. L. et al. Rank methods for combination of independent experiments in analysis of variance. **The Annals of Mathematical Statistics**, [S.l.], v.33, n.2, p.482–497, 1962.

HORNIK, K.; BUCHTA, C.; ZEILEIS, A. Open-source machine learning: r meets weka. **Computational Statistics**, [S.l.], v.24, n.2, p.225–232, 2009.

JO, T.; JAPKOWICZ, N. Class imbalances versus small disjuncts. **ACM Sigkdd Explorations Newsletter**, [S.l.], v.6, n.1, p.40–49, 2004.

KARATZOGLOU, A. et al. kernlab – An S4 Package for Kernel Methods in R. **Journal of Statistical Software**, [S.l.], v.11, n.9, p.1–20, 2004.

KHOSHGOFTAAR, T. M.; REBOURS, P. Improving software quality prediction by noise filtering techniques. **Journal of Computer Science and Technology**, [S.l.], v.22, n.3, p.387–396, 2007.

KHOSHGOFTAAR, T. M.; VAN HULSE, J.; NAPOLITANO, A. Comparing boosting and bagging techniques with noisy and imbalanced data. **IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans**, [S.l.], v.41, n.3, p.552–568, 2011.

KRAWCZYK, B. et al. Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy. **Applied Soft Computing**, [S.l.], v.38, p.714–726, 2016.

KUBAT, M.; MATWIN, S. et al. Addressing the curse of imbalanced training sets: one-sided selection. In: ICML. **Anais...** [S.l.: s.n.], 1997. v.97, p.179–186.

LAURIKKALA, J. Improving identification of difficult small classes by balancing class distribution. In: CONFERENCE ON ARTIFICIAL INTELLIGENCE IN MEDICINE IN EUROPE. **Anais...** [S.l.: s.n.], 2001. p.63–66.

LIAW, A.; WIENER, M. Classification and Regression by randomForest. **R News**, [S.l.], v.2, n.3, p.18–22, 2002.

LICHMAN, M. **UCI Machine Learning Repository**. 2013.

LING, C. X.; SHENG, V. S. Cost-sensitive learning. In: **Encyclopedia of machine learning**. [S.l.]: Springer, 2011. p.231–235.

LÓPEZ, V. et al. An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics. **Information Sciences**, [S.l.], v.250, p.113–141, 2013.

MANI, I.; ZHANG, I. kNN approach to unbalanced data distributions: a case study involving information extraction. In: OF WORKSHOP ON LEARNING FROM IMBALANCED DATASETS. **Proceedings...** [S.l.: s.n.], 2003.

MENZIES, T.; KRISHNA, R.; PRYOR, D. **The Promise Repository of Empirical Software Engineering Data**. 2015.

MICHIE, D.; SPIEGELHALTER, D. J.; TAYLOR, C. C. **Machine learning, neural and statistical classification**. [S.l.]: Citeseer, 1994.

NG, W. W. et al. Diversified sensitivity-based undersampling for imbalance classification problems. **IEEE transactions on cybernetics**, [S.l.], v.45, n.11, p.2402–2412, 2015.

PARZEN, E. On estimation of a probability density function and mode. **The annals of mathematical statistics**, [S.l.], v.33, n.3, p.1065–1076, 1962.

PRATI, R. C. et al. Class imbalances versus class overlapping: an analysis of a learning system behavior. In: MICAI. **Anais...** [S.l.: s.n.], 2004. v.4, p.312–321.

QUINLAN, J. R. **C4. 5**: programs for machine learning. [S.l.]: Elsevier, 2014.

RAUDYS, S. J.; JAIN, A. K. et al. Small sample size effects in statistical pattern recognition: recommendations for practitioners. **IEEE Transactions on pattern analysis and machine intelligence**, [S.l.], v.13, n.3, p.252–264, 1991.

RIVERA, W. A. Noise Reduction A Priori Synthetic Over-Sampling for class imbalanced data sets. **Information Sciences**, [S.l.], v.408, p.146–161, 2017.

SÁEZ, J. A. et al. SMOTE–IPF: addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. **Information Sciences**, [S.l.], v.291, p.184–203, 2015.

SCHLIEP, K.; HECHENBICHLER, K. **kknn**: weighted k-nearest neighbors. [S.l.: s.n.], 2016. R package version 1.3.1.

SCHÖLKOPF, B.; SMOLA, A. J. **Learning with kernels**: support vector machines, regularization, optimization, and beyond. [S.l.]: MIT press, 2002.

SEIFFERT, C. et al. An empirical study of the classification performance of learners on imbalanced and noisy software quality data. **Information Sciences**, [S.l.], v.259, p.571–595, 2014.

SHEPPERD, M. et al. Data Quality: some comments on the nasa software defect datasets. **IEEE Trans. Softw. Eng.**, Piscataway, NJ, USA, v.39, n.9, p.1208–1215, Sept. 2013.

TING, K. M. The problem of small disjuncts: its remedy in decision trees. , [S.l.], 1994.

TOMEK, I. Two modifications of CNN. **IEEE Trans. Systems, Man and Cybernetics**, [S.l.], v.6, p.769–772, 1976.

WANG, S.; YAO, X. Using class imbalance learning for software defect prediction. **IEEE Transactions on Reliability**, [S.l.], v.62, n.2, p.434–443, 2013.

WASIKOWSKI, M.; CHEN, X.-w. Combating the small sample class imbalance problem using feature selection. **IEEE Transactions on knowledge and data engineering**, [S.l.], v.22, n.10, p.1388–1400, 2010.

WEISS, G. M. Mining with rarity: a unifying framework. **ACM Sigkdd Explorations Newsletter**, [S.l.], v.6, n.1, p.7–19, 2004.

WEISS, G. M. The Impact of Small Disjuncts on Classifier Learning. In: DATA MINING. **Anais...** [S.l.: s.n.], 2010. v.8, p.193–226.

WEISS, G. M.; PROVOST, F. The effect of class distribution on classifier learning: an empirical study. **Rutgers Univ**, [S.l.], 2001.

WILCOXON, F. Individual comparisons by ranking methods. **Biometrics bulletin**, [S.l.], v.1, n.6, p.80–83, 1945.

WILSON, D. L. Asymptotic properties of nearest neighbor rules using edited data. **IEEE Transactions on Systems, Man, and Cybernetics**, [S.l.], v.2, n.3, p.408–421, 1972.

WOODS, K. S. et al. Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography. **International Journal of Pattern Recognition and Artificial Intelligence**, [S.l.], v.7, n.06, p.1417–1436, 1993.

YEN, S.-J.; LEE, Y.-S. Cluster-based under-sampling approaches for imbalanced data distributions. **Expert Systems with Applications**, [S.l.], v.36, n.3, p.5718–5727, 2009.

YEUNG, D. S. et al. Localized generalization error model and its application to architecture selection for radial basis function neural network. **IEEE Transactions on Neural Networks**, [S.l.], v.18, n.5, p.1294–1305, 2007.

ZHANG, H.; LI, M. RWO-Sampling: a random walk over-sampling approach to imbalanced data classification. **Information Fusion**, [S.l.], v.20, p.99–116, 2014.