Department of Computer Architecture and Technology
Konputagailuen Arkitektura eta Teknologia saila (KAT)
Departamento de Arquitectura y Tecnología de Computadores (ATC)

Universidad del País Vasco
Euskal Herriko Unibertsitatea

University of the Basque Country

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Contributions to Comprehensible Classification

Ph.D. Dissertation presented by
Igor Ibarguren Arrieta

Supervised by
Jesús María Pérez de la Fuente
Javier Francisco Muguerza Rivero

Donostia, June 2020

This work was funded by the Department of Education, Universities and Research of the Basque Government (LIPCNE, grant IT-395-10; ADIAN, grant IT-980-16; and the PREDOC grant PRE-2013-1-887, BOPV/2013/128/3067); the University of the Basque Country UPV/EHU (BAILab, grant UFI11/45) and by the Ministry of Economy and Competitiveness of the Spanish Government and the European Regional Development Fund - ERDF (Physcomp, TIN2017-85409-P).
A research stay at the College of Arts and Sciences of the American University in Washington D.C. (United States of America) was funded by the Department of Education, Universities and Research of the Basque Government (EGONLABUR EP_2016_1_0053).

"Computer Science is no more about computers than astronomy is about telescopes"
- Edsger Wybe Dijkstra

Laburpena

Memoria honetan deskribatutako doktoretza tesiak bi sailkapen algoritmo ulergarri mota ezberdinen emaitzen hobekuntzan egin ditu ekarpenak: erabaki zuhaitz kontsolidatuen algoritmoetan eta PART tipoko erregela indukzio algoritmoetan.

Erabaki zuhaitz kontsolidatuen algoritmoei egindako ekarpenei dagokienez, lehenik berlaginketarako estrategia berri bat proposatu da, non azpilagin kopurua azpilagin hauen klase banaketaren aldaketara doitzen den informazioa ez galtzeko. Estrategia hau erabiliz, C4.5 algoritmoaren bertsio kontsolidatuak (CTC) algoritmo genetikoetan oinarritutako hainbat algoritmo eta beste hainbat sailkapen algoritmo ulergarri klasikok baino emaitz hobeagoak lortzen ditu. Hiru algoritmo berri kontsolidatu dira: CHAID en bariante bat (CHAID*) eta C4.5 eta CHAID* en *Probability Estimation Tree* bertsioak (C4.4 eta CHAIC). Algoritmo kontsolidatu guztiek beraien oinarri algoritmoek baino emaitz hobeagoak lortzen dituzte. Azkenik, zuhaitzen inausketa prozesuak algoritmo sinple eta kontsolidatuengan duen ondorioa analizatu da, eta tesi honetan proposatutako inausketa estrategiak emaitz honenak lortzen dituela ondorioztatu da.

PART tipoko erregela indukzio algoritmoei egindako ekarpenei dagokienez, lehen proposamen batek PART-ek zuhaitz partzialak sortu eta hauetatik erregelak ateratzeko egiten dituen erabaki batzuk aldatzen ditu, eta ondorioz, PART ekin konparatuta, orokortzeko gaitasun hobea eta konplexutasun estruktural baxuagoa duten sailkatzaileak sortzen ditu. Bigarren proposamen batek, zuhaitz partzialak erabili beharrean, guztiz garatutako zuhaitzak erabiltzen ditu, eta oraindik orokortzeko gaitasun hobeagoa eta konplexutasun estruktural baxuagoa duten sailkatzaileak sortzen ditu. Bi proposamen hauek eta PART algoritmo originala CHAID* algoritmoan oinarritutako bertsioekin osatu dira, hobekuntza hauek zuhaitz algoritmo ezberdinei hedatu ahal diren behatzeko, eta ondorioztatu da CHAID*-en oinarritutako PART tipoko algoritmoek CHAID*-ek baino egitura sinpleago eta orokortzeko gaitasun handiagoa duten sailkatzaileak sortzen dituztela.

Resumen

La tesis doctoral descrita en esta memoria ha contribuído a la mejora de dos tipos de algoritmos de clasificación comprensibles: algoritmos de árboles de decisión consolidados y algoritmos de inducción de reglas tipo PART.

En cuanto a las contribuciones a la consolidación de algoritmos de árboles de decisión, se ha propuesto una nueva estrategia de remuestreo que ajusta el número de submuestras para permitir cambiar la distribución de clases en las submuestras sin perder información. Utilizando esta estrategia, la versión consolidada de C4.5 (CTC) obtiene mejores resultados que un amplio conjunto de algoritmos comprensibles basados en algoritmos genéticos y clásicos. Tres nuevos algoritmos han sido consolidados: una variante de CHAID (CHAID*) y las versiones *Probability Estimation Tree* de C4.5 y CHAID* (C4.4 y CHAIC). Todos los algoritmos consolidados obtienen mejores resultados que sus algoritmos de árboles de decisión base, con tres algoritmos consolidados clasificándose entre los cuatro mejores en una comparativa. Finalmente, se ha analizado el efecto de la poda en algoritmos simples y consolidados de árboles de decisión, y se ha concluido que la estrategia de poda propuesta en esta tesis es la que obtiene mejores resultados.

En cuanto a las contribuciones a algoritmos tipo PART de inducción de reglas, una primera propuesta cambia varios aspectos de como PART genera árboles parciales y extrae reglas de estos, lo cual resulta en clasificadores con mejor capacidad de generalizar y menor complejidad estructural comparando con los generados por PART. Una segunda propuesta utiliza árboles completamente desarrollados, en vez de parcialmente desarrollados, y genera conjuntos de reglas que obtienen aún mejores resultados de clasificación y una complejidad estructural menor. Estas dos nuevas propuestas y el algoritmo PART original han sido complementadas con variantes basadas en CHAID* para observar si estos beneficios pueden ser trasladados a otros algoritmos de árboles de decisión y se ha observado, de hecho, que los algoritmos tipo PART basados en CHAID* también crean clasificadores más simples y con mejor capacidad de clasificar que CHAID*.

Abstract

The doctoral thesis described in this dissertation has contributed to the improvement of two types of comprehensible classification systems: consolidated decision tree algorithms and PART-like ruleset induction algorithms.

Regarding the contributions to the consolidation of decision tree algorithms, a new resampling strategy has been proposed to automatically adjust the number of subsamples to allow changing the class distribution of subsamples without loss of information. Using this strategy, the consolidated version of C4.5 (CTC) performs better than a wide set of genetics-based and classical comprehensible algorithms. Three new algorithms have been consolidated: a variant of CHAID (CHAID*) and Probability Estimation Tree versions of C4.5 and CHAID* (C4.4 and CHAIC). All of these consolidated algorithms show better performance than their base decision tree algorithm, with three of the consolidated algorithms placing in the top four positions within the same comparison. Finally, the effect of pruning has been analyzed on simple decision tree algorithms and their consolidated versions, and it has been concluded that the pruning strategy proposed in the thesis works best.

Regarding the contributions to PART-like ruleset induction algorithms, a first proposal changes several aspects of how PART creates partial trees and extracts ruleset from them, which results in more accurate and significantly less complex classifiers than the original PART being generated. A second proposal uses fully developed decision trees, instead of partial trees, and generates even more accurate and simpler rulesets. These two new proposals and the original PART have been complemented with CHAID*-based variants to observe if these benefits can apply to other decision tree algorithms, and it has been observed that, in fact, CHAID*-based PART-like algorithms also create simpler and more accurate classifiers than CHAID*.

Eskerrak

Lehenik eta behin, eskerrik asko nire zuzendariei, Txus eta Javi, orain dela urte dexente aukera eman eta sortatzen zitzaizkidan zalantza guztiak argitzeko prest egoteagatik. Zuek biok aurrez egindako lan guztiagatik ez bazen izan, lehen urteak gogorragoak izango ziren.

Denboran zehar laborategi kideak izan zareten gutzioi Aizea, Iñigo, Ainhoa, Otzeta eta Ugaitz. ALDAPA taldeko beste kideei Olatz, Ibai, Natxo, Agus, Joseba eta Jodra, nolabait guztiok lagundu nauzue mugarri hontara heltzen.

ADIAN taldeko zuzendariari, Julio Abascal. Zure esfortzuak nire lana erreztu du urte hauetan zehar.

Jon Kepa-ri, ikerkuntzan lehen aukera eman, eta nigan interes hori esnatu izanagatik. Nirekin hasi zinenten lagunei: Maria, Zuru, Ibai, David... batzuk ni baino lehenago amaitu duzue. Batzuek gidatu egin nauzue eta beste batzuk nik gidatu ditut.

Eskerrik asko American University-ko Nathalie Japkowicz irakasleari nire egonaldian jasotzeagatik eta bertan lan egin genuen denboragatik.

Nire gurasoei, doktoretza beka onartzeko bikoitza ordaintzen zidan lanpostu bat utzi behar nuela esan nienean, ez molestatzeaz gain, lagundu egin nindutelako.

Asierri, ezagutu ginenean jada bide hau hasita neukan baina alderdi askotan lagundu didazu amaitzen. Alex-i, doktoretzan zehar bota genituen juergak garai txarretan ere momentu onak izaten lagundu zuten.

Lan honi azkeneko bultzada emateko gogaitu nauzuen guztioi.

Tesian zehar erabili ditudan tresna guztien sortzaileei: nire ikerkuntza taldeari, ALDAPA, Haritza plataformagatik, nire lanaren oinarria izan eta non algoritmo guztiak inplementatu diren; Euskal Herriko Unibertsitateko Borka Calvo eta Nafarroako Unibertsitate Publikoko Guzmán Santafé-ri *scmamp* herremintagatik; eta Granadako Unibertsitateko *Soft Computing y Sistemas de Información Inteligentes* taldeari test estatistikoak egiteko erabilitako *KEEL* tresnagatik.

Azkenik, nire lanarentzat oinarri gisa erabili ditudan lanen egileei. Azken finean, denok erraldoien sorbaldetan esertzen gara.

Agradecimientos

Lo primero, gracias a mis directores, Txus y Javi, por haberme dado la oportunidad hace años y haber estado ahí a un toque de puerta para todas las dudas que me iban surgiendo. Si no hubiese sido por todo el trabajo previo que ya habíais hecho, los primeros años hubiesen sido mucho más largos.

A los diferentes compañeros de laboratorio que habéis estado a lo largo del tiempo Aizea, Iñigo, Ainhoa, Otzeta y Ugaitz. A los demás compañeros del grupo ALDAPA Olatz, Ibai, Natxo, Agus, Joseba y Jodra, que de alguna manera u otra habéis contribuido a que haya podido alcanzar este hito.

A Julio Abascal, director del grupo de investigación ADIAN. Tus esfuerzos han facilitado el trabajo estos años.

A Jon Kepa por haberme dado mi primera oportunidad en investigación y haber despertado esa motivación en mí. A los compañeros que empezasteis más o menos a la vez que yo: Maria, Zuru, Ibai, David... algunos habéis ido más rápido que yo. Me habéis guiado y yo he podido guiar a otros.

Gracias a la profesora Nathalie Japkowicz de la American University por haberme acogido durante mi estancia y el tiempo que trabajamos juntos.

A mis padres, que cuando les dije que iba a dejar un trabajo para aceptar una beca predoctoral cobrando la mitad, no solo no les molestó, si no que además me apoyaron.

A Asier que aunque nos conocimos cuando yo ya había empezado este camino me has ayudado en muchos aspectos a terminarlo. A Alex, todas esas juergas durante los años de doctorado ayudaron a tener buenos momentos hasta en malas épocas.

A todos los que no habéis dejado de molestarme para que le pusiese el lazo a este trabajo.

A los creadores de las herramientras que he utilizado durante la tesis: mi grupo de investigación, ALDAPA, por Haritza donde todos los algoritmos han sido implementados y que ha sido base para mi trabajo; Borja Calvo de la Universidad del País Vasco y Guzmán Santafé de la Universidad Pública de Navarra por la herramienta *scmamp*; y al grupo de investigación *Soft Computing y Sistemas de Información Inteligentes* de la Universidad de Granada por la herramienta *KEEL* utilizada para los tests estadísticos.

Por último, a todos los investigadores cuyo trabajo ha servido como base para yo poder realizar el mío. Al fin y al cabo, todos nos sentamos a hombros de gigantes.

Acknowledgements

First and foremost, thanks to my advisors, Txus and Javi, for having given me the opportunity all those years ago and being there, one door knock away, for every doubt that I had. If it had not been for all the previous work done by you, the first years would have lasted longer.

To all lab colleagues over the years: Aizea, Iñigo, Ainhoa, Otzeta and Ugaitz. To the rest of colleagues of the ALDAPA research group Olatz, Ibai, Natxo, Agus, Joseba and Jodra, who in one way or another have helped me achieve this milestone.

To Julio Abascal, director of the ADIAN research group. Your efforts have helped my work this years.

To Jon Kepa for having given me my first opportunity in the field of research and having awoken that motivation in me. To the colleagues that begun more or less at the same time as I did: Maria, Zuru, Ibai, David... some of you have finished faster than I have. You have guided me and I have guided others.

Thanks to Professor Nathalie Japkowicz of the American University for hosting me during my visit and the time we worked together.

To my parents, who when I told I was leaving my job to accept a PhD scholarship and halving my income, not only did not bother them, but actually supported me.

To Asier, we met when I was already half-through this journey but you have helped me finish it in so many aspects. To Alex, all those Saturday nights during the PhD years helped have good moments even during bad times.

To everyone who has kept bothering me to finish this work.

To the creators of the different tools I have used during the thesis: my research group ALDAPA, for developing the Haritza platform that has served as base for my work and where all algorithms have been implemented; Borja Calvo of the University of the Basque Country and Guzmán Santafé from the Public University of Navarre for the *scmamp* tool; and the *Soft Computing y Sistemas de Información Inteligentes* group of the University of Granada for the *KEEL* tool used for the statistical testing.

Finally, thanks to all researchers whose work has helped me carry out mine. In the end, we all stand on shoulders of giants.

Contents

List of Figures

List of Tables

List of Algorithms

Part I

Introduction

Chapter 1

Introduction

In data mining, a classification problem occurs when an object needs to be assigned to a predefined group or class based on a number of observed attributes related to that object [159].

In some cases the reason why the classification is made is almost as important as the accuracy of the decision, thus the classifier must be comprehensible. This is especially true in domains where classification is used as a decision support system for humans, like medical diagnosis or fraud detection. Some simple classifier systems such as decision trees and rulesets have this explaining capacity.

The thesis described in this dissertation delves into two research lines about classifiers with explaining capacity: contributions to the consolidation of decision tree algorithms, and contributions to the improvement of ruleset algorithms based on the PART algorithm.

The most common way of improving the results of decision trees is to create ensemble classifiers with them. Ensembles are multiple classifier systems (MCS, of which Bagging and Boosting are the most used examples and will be described later) that combine the knowledge of multiple individual classifiers. The individual classifiers can be completely different, however, it is most usual to use the same algorithm with different samples. Ensembles achieve a greater accuracy than individual classifiers, but the complexity of their models, and how the final classification decision is made, mean that the comprehensibility of the original classifier is lost.

The consolidation methodology was conceived as a middle ground of simple comprehensible classifiers and an ensemble classifier made of them. This methodology also creates multiple samples, but applies the ensemble voting process during the classifier building phase. This allows it to use the knowledge of multiple samples, but creating a simple classifier that keeps the base algorithm's comprehensibility.

Consolidation was proposed to solve a problem of fraud detection in car insurance claims. In that case, the classifier had to be comprehensible because it was an employee of the insurance company who made the final decision about

whether to declare the claim fraudulent or not, because insurance companies usually prefer to let some fraudulent claims slip, than wrongly accusing an honest customer. Also, this particular insurance problem was representative of another issue classification algorithms face: the class imbalance problem.

Class imbalance has been considered one of the main problems in data mining in recent years [156]. This problem occurs when at least one of the classes (minority class or classes) is underrepresented in the original training sample compared to the remaining classes. Class imbalance is present in several real problems, such as medical diagnosis [106], traffic incident detection [160], DNA sequencing [68], and fraud detection [119].

Initially, consolidation created multiple subsamples that kept the dataset's original class distribution. However, recent research suggests that the original class distribution is usually not the best one. In fact, this research suggests that subsamples with a balanced class distribution might get better results. However, switching to a balanced subsample system requires implementing a new resampling strategy to reduce data loss due to excessive undersampling in heavily imbalanced datasets. This poses the following questions:

- How could information loss be avoided when greatly changing the class distribution of subsamples?
- How could the information loss be "equally fair" between datasets with very different class distributions?

Another pending work around consolidation is applying the methodology to other decision tree algorithms. So:

- Can other decision tree algorithms benefit from applying consolidation?

On the other hand, PART is a widely known ruleset induction algorithm. It is part of the WEKA platform and has been cited hundreds of times. It combines the two most-used rule induction strategies: extracting rules from trees and the separate-and-conquer strategy. It creates ruleset by extracting individual rules from partially developed C4.5 decision trees.

As in all machine learning algorithms, the decisions made during the classifier building process affect the performance of the generated models. One of the most prominent decisions PART makes is that instead of fully developing decision trees, they are partially developed. In fact, the algorithm takes the name from this feature. Partial trees are built by guiding the tree's development using a local search method based on the entropy of the nodes. According to PART's authors this is done for efficiency, expecting low entropy nodes to expand less times, thus generating shorter, more general rules. However it is known that global search methods can create more accurate classifiers, albeit at a higher complexity. This leads to the questions:

- Would using a global search mechanism improve the accuracy of PART?

- Would this significantly increase the complexity of the classifiers created by the algorithm?

These questions led to the development, during the thesis, of a PART-like algorithm that uses global search. This contribution also changes some other decisions that increase the size of the partial tree, while adding a pruning process and allowing shorter untreated leaves to be considered as rules, in order to compensate for the bigger (less general) partial trees.

An evolution of this idea would be to use completely developed decision trees to extract rules from. This would eliminate the need for the rest of seemingly arbitrary decisions PART makes: no need to use a search mechanism to drive the tree's expansion, no need to exclude some nodes for the rule, and no need to perform early pruning of trees. So the next question is:

- How would using fully developed decision trees affect the accuracy and complexity of rulesets generated by PART?

Using decision trees to extract rules from, also allows to switch the base decision tree algorithm used by PART-like algorithms (C4.5). PART usually generates more accurate classifiers than C4.5, so:

- Could another decision tree algorithm benefit from the PART-like treatment and generate better classifiers?
- Or, is this behavior only limited to C4.5?

Finally, while this thesis delves into the field of comprehensible classifiers and proposes and compares a decent size of algorithms, they are restricted to consolidated decision tree algorithms and different PART-like ruleset induction algorithms. A final question would be:

- How would all of these algorithms compare to a wider set of comprehensible rule and tree induction algorithms?

Luckily, a study comparing 22 genetics-based and classical rule and decision tree algorithms was published recently, and its results were encouraged to be used as benchmark by other members of the machine-learning research community. So all of the proposals of this thesis are compared to that wider set of comprehensible algorithms.

1.1 Structure of the Dissertation

This dissertation has been divided into five main parts: Introduction, Background Work, Contributions, Conclusions, and Appendices.

Part I, Introduction, describes the motivation behind the work carried out during the thesis, and enumerates the research questions to be answered throughout it.

Part II, Background Work, summarizes the literature on the topics that serve as base for the work carried out in the thesis. These topics include supervised learning, comprehensible classifiers, the class imbalance problem, decision trees, consolidation, the PART ruleset algorithm, and the statistical validation of the experimental results.

Part III, Contributions, details the main contributions of this dissertation, and describes the carried out experimental work. This is a three chapter part. The first two chapters are dedicated to the distinct research lines: consolidation of decision tree algorithms, and improvement of PART-like ruleset algorithms. Chapter 3 focuses on contributions made to the consolidation of decision tree algorithms: a new resampling strategy, an extension of the consolidated algorithms, and experimental work on the effect of pruning on consolidated trees. Chapter 4 presents two new PART-like algorithms: BFPART and UnPART. These algorithms change how PART-like rulesets are generated and improves their results from the points of view of the ability to generalize and structural complexity. Chapter 5 combines the results from the previous chapters to perform a global analysis.

Part IV, Conclusions, is composed of a single chapter, Chapter 6. This chapter outlines the conclusions drawn from the contributions of the thesis, proposes some future work, and lists the publications that have resulted from the work carried out in this thesis.

Part V consists of a series of appendices that extend the information presented in Part III.

Part II

Background Work

Chapter 2

Background Work

This chapter will provide the necessary context to understand the work carried out throughout this thesis. The first two sections describe the subfields of machine learning research this work belongs to: namely the machine learning subfield of supervised learning or classification, and the specific field of comprehensible classifiers. The following section describes the problem of class imbalance in machine learning, a specific problem tacked by the work presented in this thesis. Sections 2.4 through 2.7 give an insight into the paradigms and algorithms that serve as base for the algorithms developed in this thesis. Finally, Sections 2.8 and 2.8.3 explore different options to evaluate the proposed work against already existing solutions.

2.1 Supervised Classification

Machine learning is a subfield of artificial intelligence that encompasses systems capable of learning concepts from data [45]. These systems create data-driven models (a structured representation) instead of using explicitly programmed models [124].

With the proliferation of information systems and geographically distributed companies and institutions, almost all of their data is transmitted (and stored) digitally. Having all of this available raw data has motivated them to squeeze as much information as possible from it, be it from an economic, self-improvement, social advancement, or another motivation.

Most learning algorithms use similar structures of data: arrays or vectors of characteristics. These characteristics are also called *features*, *attributes* or *variables*. These variables can be dependent and independent. Learning algorithms work under the assumption that the values of independent variables influence the value of the dependent variables. Depending on how many values a variable takes, it is treated differently. On the one hand, if an attribute has a finite set of values, it is considered a discrete attribute. Within *discrete* variables, another subdivision is made. If the values the variable takes have an intrinsic

order (number of children, level of studies), they are considered *ordinal* values, and if there is no order between the values (color, ethnicity, gender) the variable is considered *nominal*. On the other hand, if the attribute can take an infinite number of values, it is a *numeric* or *continuous* variable (physical measures, age). A single vector of data represents a specific case, which can also be called an *example* or *instance* [124] using the technical language. Multiple instances representing the same problem form a *dataset*. Learning algorithms are used to train models. A model is a representation of the underlaying patterns present in the data. Training a model means analyzing the already available data looking for patterns, and then creating a system that is capable of processing previously unseen data by means of this patterns.

Learning algorithms are usually divided into two main groups: *supervised learning* (also known as classification) and *unsupervised learning* (also known as clustering). The difference between classification and clustering systems is that in classification, during the training, the value of the dependent variable is known; the input information is labeled. In clustering, the input data is not labeled, and the algorithm works by finding similarities between instances and grouping them in different clusters according to these similarities. Within supervised learning, if the dependent variable is discrete, we face a classification problem, whereas if the dependent variable is numeric, it is a regression problem. In classification problems, the dependent variable is referred to as the *class*. Depending on how many values the class can take, it is either a two-class problem, or a multi-class problem. Models built by supervised learning algorithms are also called *classifiers*. Both supervised and unsupervised learning techniques have been successfully applied on a wide variety of fields such as spam filtering [130], detection of counterfeit goods [123], software fault prediction [28], suicide risk assessment [41], user behavior modeling [13], computer vision [60], and recommendation systems [139]. Table 2.1 shows the structure of a dataset for classification with n instances, m independent features, and the *class* dependent feature which can take k values.

Instance	V_1	V_2	V_3	...	V_m	class
I_1	x_{11}	x_{12}	x_{13}	...	x_{1m}	C_3
I_2	x_{21}	x_{22}	x_{23}	...	x_{2m}	C_1
I_3	x_{31}	x_{32}	x_{33}	...	x_{3m}	C_k
...	...	...	...	...	...	...
I_n	x_{n1}	x_{n2}	x_{n3}	...	x_{nm}	C_2

Table 2.1: Example of a dataset's structure.

A dataset is considered noisy when training data seems to have the incorrect class set [2]. This situation can be easily identified if two otherwise exact instances differ in their class value. A similar situation would be when examples of

one class are completely surrounded by examples of a different class. Noise can affect the ability of algorithms to correctly separate classes, lead to overfitting [155], and increase complexity of the classifiers [127]. Worst-case scenario, this could be because the attributes present in the domain are not enough to correctly define the case. However, it is much more likely that data was incorrectly collected or manually entered.

Sometimes, an instance does not have a set value for one or multiple variables. This is what is called a *missing* value. Missing values can happen for a variety of reasons: data was not collected or entered for that instance, incorrect data handling... This value usually needs a special treatment and most algorithms have a way of coping with it.

2.2 Comprehensibility and Comprehensible Classifiers

Among the many uses for classifiers, an explanation of how a decision is made is not needed. This is usually because consequences for incorrect classifications are not very harsh, or the problem is so well validated that the system's decision is trusted [46].

However, in one hand, there are fields where a classifier does not make a final decision, and is used as a decision support system for a human user. This is prevalent in fields where decisions affect human lives, such as medicine and finance. In these cases, the reasons why a decision is made are almost as important as the accuracy of the decision [38, 95, 69], as these reasons will help the human user reach a decision. On the other hand, when a new machine learning system is being built, being able to represent the decision making mechanisms of a model can help human users trust the machine learning model [126, 162].

In recent years, the comprehensibility of the knowledge extracted from classifiers has been an area of growing interest in the machine learning community. Several special sessions have been held in multiple important conferences[1][2] [3][4][5][6][7].

[1]Towards interpretable ML applications in biomedicine and health at IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI 2014)

[2]Interpretable systems in machine learning, data analysis, and visualization at IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2013) within the IEEE Symposium Series on Computational Intelligence (SSCI 2013)

[3]Workshop on machine learning and interpretation in neuroimaging at Neural Information Processing Systems (NIPS 2011, 2013, 2014)

[4]Interpretable models in machine learning at the 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2012)

[5] Interpretability in Computational Intelligence Systems at IEEE World Congress on Computational Intelligence (WCCI 2012)

[6]Interactive Data Analysis and Visualization at IEEE International Joint Conference on Neural Networks (IJCNN 2012)

[7]The importance of visualization in real-world machine learning applications at the 19th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2011)

The "Special Issue on Interpretable Fuzzy Systems" editorial [10] was recently published on the Information Sciences journal, where besides defining interpretability, the authors emphasized on its importance on a wide variety of scientific fields (beyond fuzzy systems). This editorial also quantified the number of articles published in this area in recent years by looking up keywords that begin with interpretab*, understandab*, comprehensib*, intelligib*, etc. on the Thompson Corporation's ISI Web of Knowledge, and found a total of 59,484 records, of which only 262 related to fuzzy systems. Finally, they highlighted the computed h-index (21), the number of quotes (2095), and the average cites per item (8), were quite high.

On the one hand, some of the most widely used and best performing classification algorithms do not produce comprehensible classifiers. Algorithms such as artificial neural networks (ANN), statistical classifiers, support vector machines (SVM), and ensembles are opaque (non-comprehensible) because, either due to their structure or complexity, work like black boxes. There is a trade-off between the performance and the comprehensibility of classifiers [47]. Even so, effort has been made to give explaining capacity to opaque algorithms, and there are several approaches to extract knowledge from non-comprehensible classifiers. This is referred to as post hoc explainability. Post hoc explainability can either be model-agnostic or model-specific. Model-agnostic approaches are more widely used because they are not restricted to the machine learning models they can extract knowledge from. Some of the simplest post hoc approaches work by producing a relevance of features instead of offering full explainability, and rank or measure the influence each input feature has in the prediction. SHAP [102] (SHapley Additive exPlanations) is one of the most widely used examples of explanation by feature relevance. Another approach of model-agnostic post hoc explainability is the explanation by simplification. These approaches create simpler models based on the outcome of the model. These surrogate models have a similar classification behavior as the black-box classifier they are based on, but are usually created using a comprehensible classifier algorithm. Domingos, the author of one such approach [43], favored simple comprehensible classifiers over multiple classifier systems by stating that "while a single decision tree can easily be understood by a human as long as it is not too large, fifty such trees, even if individually simple, exceed the capacity of even the most patient user".

On the other hand, in the literature, several machine learning algorithms are considered to produce transparent models. Transparent models are understandable by themselves. Decision tree and ruleset induction algorithms, for example, are considered to produce transparent models. These algorithms generate comprehensible classifiers because their models are successions of conditions related to the variables and their values, easily interpreted by a human [80].

Even though several decision tree algorithms (ID3 [127], CART [22], CHAID [98], etc.) have been designed (a review can be found on [110]), the C4.5 algorithm designed by Ross Quinlan is probably the most widely used algorithm with explaining capacity, and it was selected as one of the top 10 data mining algorithms at the IEEE International Conference on Data Mining held in 2006 [154]. This algorithm will be described in greater detail later in this chapter as

it the base algorithm for some algorithms proposed in this thesis.

Regarding ruleset induction algorithms Quinlan himself proposed a strategy (C4.5rules) [128] that extracts rules from C4.5 trees to make them even more comprehensible. Apart from the strategy of extracting rulesets from decision trees, there is another ruleset induction paradigm known as separate-and-conquer [61], where sequentially, rules are built to cover a subset of the training set, the subset already covered by existing rules is removed, and more rules are created using the remaining training set. Following this strategy, Cohen proposed RIPPER [36], an algorithm that achieved competitive results in regards of generalization, but being more computationally efficient than C4.5rules. The PART ruleset algorithm [55] combines both strategies: it extracts rules from decision trees and performs separate-and-conquer. This algorithm also serves as base for others developed in this thesis and will be described in more detail later in this chapter.

The most straightforward way of improving the performance of decision trees and rulesets is to use them in ensemble classifiers. Ensembles, however, lose their transparency. One approach to improving the performance of decision tree and ruleset induction algorithms while keeping their transparency is to use evolutionary algorithms to induce them. In [53], the authors proposed a taxonomy for Genetics-Based algorithms for Machine Learning (GBML) for rule induction, and carried out an extensive study. Moreover, in [62], the authors carried out a review of interpretability measures used in linguistic fuzzy rule-based systems.

In [62], the authors stated that interpretability is a subjective property, and that the choice of adequate metrics is still an unresolved issue. Decision trees and rulesets share a similar representation and can be represented as a set of conditions and a class assignation for instances covered by these conditions. The complexity of the models generated by these algorithms can be measured as the number of conditions in the ruleset or the decision tree, the number of conditions in each rule or tree branch, the number of rules or branches, etc. Rulesets with more antecedents and decision trees with more nodes might perform better, but at some point, if a model grows too much, while staying theoretically comprehensible, it becomes too complex for a human user to understand.

2.3 The problem of class imbalance

The class imbalance problem occurs when one of the classes (minority class) is under represented in the original training sample compared to the rest of the classes (majority class). When the imbalance is directly related to the nature of the data, this imbalance is considered *intrinsic*, and *extrinsic* when it is not. An example of intrinsic imbalance is the diagnosis of rare diseases where the positive cases (minority class) are overwhelmingly exceeded by the amount of negative cases (majority class). Extrinsic imbalance can be caused by limitations in the data collection process [33]. Class imbalance is present in several real problems, such as medical diagnosis [106], insurance fraud detection [119], customer churn

prevention [24], traffic incident detection [160] and DNA sequencing [68]. This problem has been considered one of the main problems in data mining in recent years [156, 79, 63, 144, 18].

When using classifiers with imbalanced data sets, the constructed model offers a high accuracy for the majority class but a really small accuracy for the minority class [79]. A classification model should have a balanced accuracy rate for all classes, or even skewed in favor of the minority class since some domains, such as the diagnosis of rare diseases, where a false negative can have worse consequences than a false positive.

Relative imbalance occurs when the examples of the minority class are greatly outnumbered by the majority class but the minority class still is fairly represented. For example a dataset with 500,000 examples and a 1:100 contains 5,000 positive examples. According to some studies, in some cases of relative imbalance the minority class is accurately learned [79, 16, 91, 150]. This suggests that imbalance is not the main deteriorating factor, but it amplifies the effect of other factors such as dataset complexity [79], small disjuncts [94], minority hubs [144], and noisy datasets [133]. A study by Japkowicz [90] indicates that the effect of class imbalance on classifiers is augmented by the concept complexity. The less complex domains, those linearly separable, do not seem to be affected by class imbalance. However, as concept complexity increases, so does the effect of the imbalance.

The class imbalance problem can seriously hamper the results of classical classifier systems. For example, a study by Chawla [30] showed that the pruning process of C4.5, aimed at removing branches that are too specialized in order to avoid overfitting, can remove branches that are key on identifying the minority class. The small size problem, where small sample size and high dimensionality occur, combined with imbalance in the dataset poses a new challenge [79].

The approaches taken to solve class imbalance can be divided in two main groups, data approaches [33, 63] and algorithmic approaches. Data approaches consist of undersampling or oversampling methods in order to balance the class distribution of the training sample. Algorithmic approaches, on the other hand, propose changes to already existing algorithms.

2.3.1 Data Approaches

Research shows that usually there is a class distribution other than the one originally found in the data sample that yields better results [150, 4]. This section describes some of the most widely used data approaches to solve the class imbalance problem. These approaches either undersample or oversample the data change the class distributions and reduce or completely remove the class imbalance.

The simplest resampling algorithms are random undersampling and random oversampling.

Random undersampling (also known as Random Subsampling) simply removes instances of a class or classes to change the class distribution in favor of other classes. In order to solve class imbalance problems instances of the ma-

jority class or classes are removed to achieve a class distribution with a higher percentage of minority class examples. Its key advantage over most of other undersampling methods is the low computational cost.

On the other hand, random oversampling randomly duplicates instances of one or more classes to also achieve a class distribution that is more balanced. For class imbalance problems this means duplicating instances of the minority class. While this approach does improve a classifier's results in some cases, it is not a widely used method because it does not generate any new information for the classification algorithm to use.

The random approaches discussed in the paragraphs above have the advantage of being fast and simple because they process instances individually without looking at the relationships between the processed instance and the rest. However, new intelligent approaches exist that, at a higher computational cost, oversample or undersample by looking at the relationships of different instances.

SMOTE or Synthetic Minority Over-sampling Technique [31] is an oversampling technique where, based on already existing minority class examples, new synthetic minority class examples are created. To achieve this, a minority class example (the reference example) and one or more of its nearest minority class neighbors are selected (five neighbors by default). For each of the continuous feature a random value between the reference instance and one of the neighbors is used. For nominal values either the reference instance's or the neighbor's value is assigned to the synthetic example. This process creates a new example within the space formed the minority class example and its neighbors. The number of minority class examples and the number of neighbors used depends on the amount of oversampling needed. SMOTE is one of the most widely used techniques to tackle class imbalance and this has resulted in several variants [76, 100, 67, 129] and combinations with other resampling techniques [16]. SMOTE has also been integrated within machine learning algorithms [29, 48].

EUS [66] or Evolutionary UnderSampling approaches use genetic algorithms to guide the search for the optimal subsample. These approaches do not seek to reach a specific class distribution, but can be configured so that only the majority class or classes are undersampled while the minority class or classes are left intact. These approaches have shown to achieve great results [101], but as it is customary for genetics-based systems, at the cost of a much greater computational cost.

2.3.2 Algorithm Approaches

This section discusses some learning algorithm approaches exclusively designed to solve the class imbalance problem.

Joshi et al. [96] proposed Modifications to Boosting algorithm, changing the weight adjusting scheme to, in case of misclassification, increase the weight of minority class examples more than the weight of majority class examples.

One-Class Learning methods use mainly, even exclusively, instances from the minority class rather than using several classes and having to learn the bound-

aries between them. Examples of one-class learning include Kernel-based One-Class classifiers [163] and One-Class classification via Neural Networks [105].

Other algorithm approaches include modifications to Support Vector Machines [153] and Evolutionary Rule-Based Systems [111].

The following subsections describe two subsets of algorithm approaches. The first subsection mentions some approaches that incorporate resampling into the algorithm, whereas the second subsection describes algorithms that work following the notion of cost.

2.3.2.1 Combination of algorithm and data approaches

The approaches described in this section combine a Multiple Classifier System (MCS) and a data approach. The most widely known data approaches, which will be more deeply described in a later section, are Bagging and Boosting. These combination approaches use one of these two MCS systems, which use multiple samples, and incorporate a resampling technique into the sample creation mechanism. SMOTEBoost [29] is a combination of the SMOTE procedure and the Boosting algorithm. It proposes a modification to Boosting where SMOTE is used on each round to create new minority class examples before applying the base algorithm. RUSBoost [136] follows the same idea but applies random undersampling between boosting iterations. SMOTEBagging [148] creates samples for Bagging using SMOTE instead of creating bootstrap samples.

2.3.2.2 Cost-Sensitive Learning

Most classification algorithms implicitly assume all classification errors have equal cost, but as it has been explained in the introduction chapter of this thesis, depending on the domain, some types of error can have worse consequences. Cost-sensitive learning takes into account the different costs of each type of misclassification and adjusts its classification criteria accordingly [50].

While it is possible to modify regular classification algorithms to make them cost sensitive, Domingos [44] proposed a procedure, MetaCost, to turn a variety of error based classifiers into cost-sensitive ones. MetaCost works by relabeling examples from the training set, based on the classification probabilities given by a classifier, to their optimal value to minimize the total cost.

Zadrozny et al. [157] extended Domingos' idea. The first modification was the removal of the assumption that the same type of misclassification carries the same cost for each example. Also, decision trees were used to estimate probabilities instead of the Bagging variation used by Domingos.

2.4 Decision Trees

Decision tree algorithms are perhaps one of the simplest supervised learning paradigms. The simplicity of the models, the availability of different implementations, the ability to explain how the classification is performed, and the

possibility of naturally being represented graphically are factors that have contributed to their popularity [142]. Most decision tree algorithms follow the foundations set by Hunt et al. [84] when they proposed Concept Learning Systems (CLS), described in Algorithm 2.1.

Input:
S: training data set
Function `expand`(S)**:**
  if *S is empty* **then**
    turn node into leaf
    `/* assign label to node.  dependent of the agorithm. */`
    return
  end
  if *all instances on S are members of class C_j* **then**
    turn node into leaf
    label with class C_j
    return
  end
  select test T with mutually exclusive outcomes $O_1, O_2, O_3, \ldots, O_n$
  partition S into subsets $S_1, S_2, S_3, \ldots, S_n$, where S_i contains instances of S that have outcome O_i for the test T
  foreach *S_i in S* **do**
    `expand` (S_i)
  end
  return

Algorithm 2.1: Basic algorithm to build decision trees.

Decision trees work by first putting the entire training set into an initial *root node*, and recursively splitting it into subsets according to the best possible split based on one of the independent attributes until any of the stopping criteria is met. Terminal nodes are called *leaf nodes* and any node between the root and a leaf node is called an *internal node*. Non-leaf nodes split the dataset into their children node according to a variable and the values this variable can take. The classes of training instances falling on each leaf node determine the class probabilities assigned to unseen examples falling on that leaf. Figure 2.1 shows an example of decision tree graphically represented. When a new example is to be classified, the classifier will check the values of the independent variables of the example, traveling along the tree from a node to one of its children, until a leaf node is reached, and the example is labeled as the majority class of the instances in a leaf.

The most obvious stopping criterion for decision trees is reaching an homogeneous node, where all instances belong to the same class. However, reaching a tree with purely homogeneous leaves is not necessarily optimal. This kind of tree usually fits so well to the training set, that in fact, it *overfits* and fails to correctly classify new unseen examples [142]. Thus, it is preferred to build parsimonious decision trees, simpler models that can competitively classify the

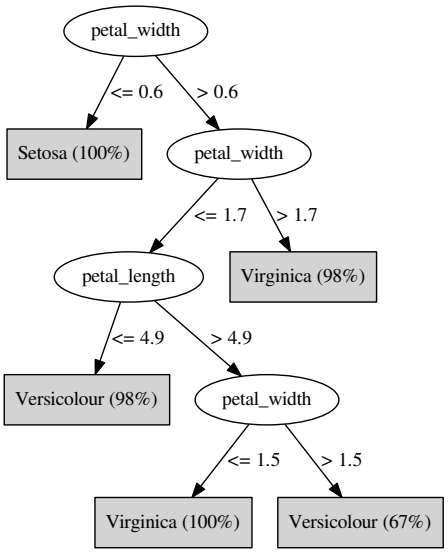

Figure 2.1: An example of a decision tree.

training set and keep the ability to generalize and correctly classify new examples. In decision trees, this is achieved by pruning the trees, either by stopping the decision tree building process early (pre-pruning), or by doing it after the tree is built (post-pruning). Pre-pruning has the advantage of saving time by avoiding to unnecessarily build a large tree that will later be pruned. However, the right thresholds for pre-pruning are difficult to achieve: a threshold that is too high will stop divisions too early before beneficial splits are found, and a value that is too low will result in little to no benefit. When any kind of pruning is applied, this most usually results in heterogeneous leaves where not all instances are of the same class. This in turn results in class probabilities being assigned to leaves, instead of class labels. These class probabilities indicate the chance of an unseen instance belonging to each class.

The following sections describe two of the most widely known and used decision tree algorithms: C4.5 and CHAID.

2.4.1 The C4.5 algorithm

C4.5 [128] is one of the most widely known decision tree induction algorithms. It is based on ID3 [127], an earlier algorithm by the same author, Ross Quinlan.

This algorithm uses the *gain ratio*, detailed in Section 2.4.1.1, as the split criterion, by testing all possible splits on each node and using that with the highest gain ratio. It is able to handle both discrete and continuous variables.

For discrete variables, by default, it creates a child node for each value the variable can take, although it is possible to configure the algorithm to look for the best subsets of values and group multiple values in the same branch. For continuous variables, it creates a binary split by looking for the best cutting value (split point), among all the values the analyzed variable takes in the training sample, and putting instances with a value smaller or equal on one branch, and those with a greater value in the other.

The C4.5 algorithm is able to handle missing values in the data. When a value for a variable is missing, C4.5 first computes the split without taking instances with a missing value into account, and if that variable is chosen for the split, the instances with missing value are put on all children's subset, adjusting their weight with the proportion of known-value instances on each branch.

C4.5 does not build trees until the training error is zero. It has two pre-pruning conditions and if any of them is met, the node is no longer split. The first of it is the node's size. For a node to be considered for a split, the sum of the instance's weights has to be equal or greater than 2 times the minimum size of a node (2 by default). The second stop condition is the value of the gain ratio. When the split with the greatest gain ratio is found, for this split to be valid, its information gain (the change in entropy) must be greater than the average information gain of all valid splits. The algorithm for C4.5 can be found on Algorithm 2.2. This algorithm uses several functions:

- `threshold` finds the best splitting threshold for continuous variables. All possible values are taken into account

- `new_test` creates possible splits for each node. For discrete variables it creates one branch per value. For continuous variables with instances with a value lower or equal to t falling into the first branch, and those with a greater value in the seonc branch.

- `average_gain_ratio` computes the average gain ratio achieved by all possible tests in T

- `best` finds the best split (greatest gain ratio) among all possible tests in T

- `gain_ratio` extracts the gain ratio value of the best split T_b

While not a stop criterion, C4.5 also possesses a pruning technique integrated in the tree building process. Once all of an internal node's children have been processed, if all children are leaves, it is tested whether replacing this internal node with a leaf would, based on training instances, increase the error more than a pre-established amount. If the error does not increase, the subtree is replaced with a leaf. This operation is called subtree replacement. Once the tree is built, C4.5 puts the recently built tree through a post-pruning process described in Section 2.4.1.2.

```
Input:
S: training data set
V: independent variables
Function expand_C4.5(S,V):
    if S is empty then
        turn node into leaf
        assign S same label as parent
        return
    end
    if S meets stop criteria then
        turn node into leaf
        assign S label of most common label of instances in S
        return
    end
    if all instances on S are members of class C_j then
        turn node into leaf
        assign label C_j to S
        return
    end
    foreach V_i in V do
        if V_i is discrete then
            T_i = new test(V_i)
        end
        if V_i is continuous then
            threshold = best_split(V_i)
            T_i = new test(V_i, threshold)
        end
        T = T ∪ T_i
    end
    gr_avg = average_gain_ratio(T)
    T_b = best(T)
    if gain_ratio(T_b) < gr_avg then
        turn node into leaf
        assign S label of most common label of instances in S
        return
    end
    foreach instance I_j in S do
        foreach outcome O_i in T_b do
            if O_i == I_ji then
                S_i = S_i ∪ I_j
            end
        end
    end
    foreach S_i subsets in S do
        expand_C4.5(S_i, X)
    end
    return
```

Algorithm 2.2: C4.5 decision tree construction algorithm.

C4.5 has been widely used in the machine learning community, both as a standalone algorithm or as base classifier for Multiple Classifier Systems [56, 21]. It has been used to extract knowledge from classifiers without explaining capacity [43], to handle data with uncertain information [145], and in combination with new intelligent resampling methods designed to tackle the class imbalance problem [16, 101].

In [65], Garcia and Herrera proposed very powerful statistical tests to analyze the significance of the differences in results, and comparing the results of five widely used algorithms (C4.5, Naive-Bayes, CN2, 1-NN, and a Kernel Classifier) over 30 datasets. Results showed that C4.5 was the most effective algorithm, followed by Naive-Bayes without significant differences, but performing significantly better than the rest of algorithms. Moreover, C4.5 was also included in an extensive study [53] detailed in Section 2.10, where it placed in the top four positions, never performing significantly worse than the best classifier.

2.4.1.1 Gain ratio

The ID3 algorithm C4.5 is based on used a criterion based on information theory: information gain.

Let $freq(C_j, S)$ be the number of instances found in the set S that belong to the class C_j, and $|S|$ be the size of the set, so that the following

$$\frac{freq(C_j, S)}{|S|}$$

represents the probability of a randomly selected instance belonging to class C_j. According to information theory, the information carried by a message can be measured in bits as

$$-\log_2\left(\frac{freq(C_j, S)}{|S|}\right).$$

So, in order to measure the information needed to identify the class of an instance in the training set S (the entropy of S), we sum the information for each class by their weight on training set S

$$info(S) = -\sum_{j=1}^{k}\left[\frac{freq(C_j, S)}{|S|} \times \log_2\left(\frac{freq(C_j, S)}{|S|}\right)\right].$$

To measure the information after subset S is divided into n subsets according to the split T, the weighted sum for the entropy of each subset has to be calculated as

$$info_T(S) = \sum_{i=1}^{n}\left[\frac{|S_i|}{|S|} \times info(S_i)\right].$$

The gain measures the information gained by partitioning a set S according to a specific split T.

$$gain(S) = info(S) - info_T(S)$$

An algorithm that uses gain aims to split the set in a way that maximizes the information gain. However, Quinlan noted that this criterion has a bias in favor of splits with many subsets. If a node splits a training set using a unique identifier for each instance, this will result in as many subsets as instances. This split achieves the maximum possible information gain. However, from a classification point of view, this split is useless. The gain ratio criterion normalizes the information gain using the split info. The split info is calculated in the same way as the entropy, but instead of measuring it from the point of view of class membership, from a point of view of membership to the subsets generated by the split.

$$split\_info(T) = -\sum_{i=1}^{n} \left[\frac{|S_i]}{|S|} \times \log_2 \left(\frac{|S_i]}{|S|} \right) \right],$$

so that the resulting criterion

$$gain\_ratio(T) = \frac{gain(T)}{split\_info(T)}$$

measures the information relevant to the classification gained by the split. C4.5 decides the variable (and value if necessary) of the split on a node by selecting the split with the highest gain ratio.

2.4.1.2 Pruning

Once the decision tree has been built, the C4.5 algorithm performs a pruning operation on the tree. While building the tree, C4.5 already performs a collapsing operation based on the training data. However, the error on training data is not a useful measure as any pruning will result on an increase in error on training data. So the key here is to try and predict errors on unseen data. Older pruning techniques such as *cost-complexity pruning* and *reduced-error pruning* held part of the data from the training set so that it could be used to assess these error dates.

The approach used by C4.5 only uses training data but errors are estimated statistically. Quinlan refers to this estimation as very pessimistic. Each leaf covers N examples, and E of them, those not belonging to the majority class on that leaf, are classified incorrectly. Using these two values, and a preset confidence level (CF, 25% by default), the errors are predicted using only the upper limit (thus, the pessimistic nature of this estimation) for the binomial distribution as $U_{CF}(E, W)$ where E is the weight of instances by the node and W is the total weight of the nodes. The upper and lower limits for the binomial distribution are symmetrical, so the probability of the actual error value being above this estimate is $CF/2$.

When pruning a tree, the whole tree is processed subtree by subtree. The algorithm first lowers down to the smallest possible subtree (internal node). Once a subtree is selected two different operations are considered:

1. Just like in the collapsing procedure. The possibility of replacing the current node with a leaf is considered. If this replacement increases the estimated errors in less than 0.1, the subtree is replaced with a leaf.

2. If the first operation fails, the algorithm considers replacing the subtree with the most populous branch. So the algorithm first computes what the error for that branch would be if the entire set of the internal node (instances falling on the most populous branch and its siblings) was put on that branch. Again, if the error in the most populous branch surpasses the error of the subtree in less than 0.1, the subtree is replaced with the biggest branch. This operation is called *subtree raising*.

The pruning algorithm is explained in Algorithm 2.3. This algorithm uses the method `largest_branch` to determine which of its children receives the most instances after the split.

Input:
N: node to be pruned
raising: whether *subtree raising* is enabled
Function `prune`$(N, raising)$**:**
  if N *is not leaf* **then**
    for *all* Ni *subnodes of* N **do**
      `prune`(N_i)
    end
    E = weight of instances incorrectly labeled by N
    W = weight of all instances in N
    $error_{leaf} = W \times U_{CF}(E, W)$
    $error_{tree} = 0$
    for *all* N_i *subsets of* N **do**
      E_i = weight of instances incorrectly labeled by N_i
      W_i = weight of all instances in N_i
      $error_{tree}$ += $|W_i| \times U_{CF}(E_i, W_i)$
    end
    if $error_{leaf} < error_{tree} + 0.1$ **then**
      turn node into leaf
      return
    end
    $j =$ `largest_branch`(N)
    $error_{branch} = W \times U^j_{CF}(E, W)$
    if $error_{branch} < error_{tree} + 0.1$ *and raising* == *true* **then**
      assign subnodes of N_j as subnodes of N
      `prune` $(N$,*raising*$)$
      return
    end
  end
  return

Algorithm 2.3: C4.5's pruning process.

2.4.2 The CHAID algorithm

The CHAID (Chi-squared Automatic Interaction Detector) algorithm [98] was the final result of several works carried out separately at different times and by different authors [108, 97]. It is currently widely used because it is available in extensively used analytical software packages such as SPSS [103] and KnowledgeSEEKER [11].

This algorithm only handles discrete variables. Unlike C4.5, this algorithm, by default, does not create a branch for each value a variable can take, and instead attempts to group multiple values on each branch. For each variable, on each node, it first creates a contingency table with the relationships between the values the variable takes and the class membership numbers. One such example is shown on Table 2.2. In this example the variable can take n values and there are k classes in domain. In this table, o_{21} would represent how many of the examples belonging to C_2 have the value x_1 for this variable. This table is passed on to Kass' algorithm [97]. This algorithm finds the most significant split for a variable according to Pearson's chi-squared test [113]. Once the most significant split for every one of variables has been found, the p-value is adjusted using the Bonferroni coefficient. The splits for all variables are compared looking for the most significant, which is used as split on that node, as long as this split is significant enough (p-value < 0.05). CHAID's algorithm can be found on Algorithm 2.4.

The algorithm developed by Kass, identified in Algorithm 2.4 by `kass`, attempts to combine values by first making the least significant binary groupings. This algorithm receives a contingency table like the one shown in Table 2.2, specific for that node, created by the function `contingency_table`. A grouping is not significant if the class distributions of the variable values (columns in the contingency table) are not significantly different (p-value > 0.05 by default) according to the χ^2 test. When combining values, it first checks the significance of every possible pair. Once groups of three or more values have been found, it tries to find the most significant binary splits (p-value ≤ 0.05 by default) by taking one of the possible values from the merge. This process iterates, while updating the contingency table with each change, until it converges in the best split for that variable, if any. Once the best split for a variable has been found, the p-value of Pearson's test is adjusted using Bonferroni coefficient. This is done for every variable. The adjusted p-values for each variable are compared

	x_1	x_2	...	x_n
C_1	o_{11}	o_{12}	...	o_{1n}
C_1	o_{21}	o_{22}	...	o_{2n}
...	...	...	...	...
C_k	o_{k1}	o_{k1}	...	o_{kn}

Table 2.2: Example of a contingency table.

Input:
S: training data set
V: independent variables
Function `expand_CHAID(`S, V`):`
  if S *is empty* **then**
    turn node into leaf
    assign S same label as parent
    return
  end
  if S *meets stop criteria* **then**
    turn node into leaf
    assign S label of most common label of instances in S
    return
  end
  if *all cases on* S *are members of class* C_j **then**
    turn node into leaf
    assign label C_j to S
    return
  end
  foreach V_i *in* V **do**
    $table_i =$ `contigency_table`(S, V_i)
    $T_i =$ `kass`$(table_i, V_i)$
  end
  $T =$ `best`(T_i)
  if $pval(T) > 0.05$ **then**
    turn node into leaf
    assign S label of most common label of instances in S
    return
  end
  foreach *instance* I_j *in* S **do**
    foreach *outcome* O_i *in* T **do**
      if $O_i = II_{ji}$ **then**
        $S_i = S_i \cup I_{ji}$
      end
    end
  end
  foreach S_i *in* S **do**
    `expand_CHAID` (S_i)
  end
  return

Algorithm 2.4: CHAID decision tree construction algorithm.

and if the split with the lowest p-value is significant and does not meet any pre-pruning criteria, the node is split and the process is repeated with each child node. The method `best` selects the best of the proposed T splits, the one with the lowest p-value, and the method `pval` simply returns the p-value of a split.

Unlike C4.5, CHAID makes a distinction between discrete variables where the values follow an established order (ordinal variables, e.g. age ranges) and variables where there is no order (nominal variables, e.g. colors). When ordinal variables are processed, only groupings that comply to this order are taken into consideration.

When looking for a split using a specific variable, if examples are found with a missing value for that variable, these examples are not used to find the best split. Once the best split is found, *missing* is considered another value the variable can take, and the algorithm attempts to merge this value with an already existing value (or value grouping). It attempts to merge the missing values with every group of the best split. The least significant merge (p-value > 0.05) will be performed, but only if it does not make the merge significant (p-value ≤ 0.05). If it is not possible to merge the missing value with others, examples with missing value are put on a branch of their own.

Regarding the pruning procedures found on CHAID, this algorithm only applies pre-pruning strategies: two strategies related to the size of the nodes and one related to the significance of the split.

- If a node has less than a certain number of instances, the algorithm does not attempt to split the node.
- If a split creates nodes with less than a certain number of instances the split is not considered valid.
- If the most significant split (split with the lowest p-value) is not significant enough (p-value < 0.05 by default) no split is performed in that node.

2.5 Ensembles classifiers

Ensemble classifiers are one of the improvements to classifiers, decision trees among others, and work by combining the decisions of several individual classifiers. Ensembles make use of data resampling, multiple feature subsets, non-deterministic algorithms and different base algorithms. Ensemble classifiers usually give better results than the classifiers they are made of [42] as long as the individual classifiers that compose the ensemble disagree with each other [77].

One of the weaknesses of decision trees, is that they are very sensitive to their training sample. A small change in the training sample can lead to the construction of a completely different tree using the same algorithm. This property is called instability [32] and unstable classifiers are referred to as weak classifiers in the context of MCS. In the case of decision trees, ensembles improve the accuracy and stability of individual trees at the cost of losing their explaining capacity. Figure 2.2 shows how an ensemble works. It shows a two class problem

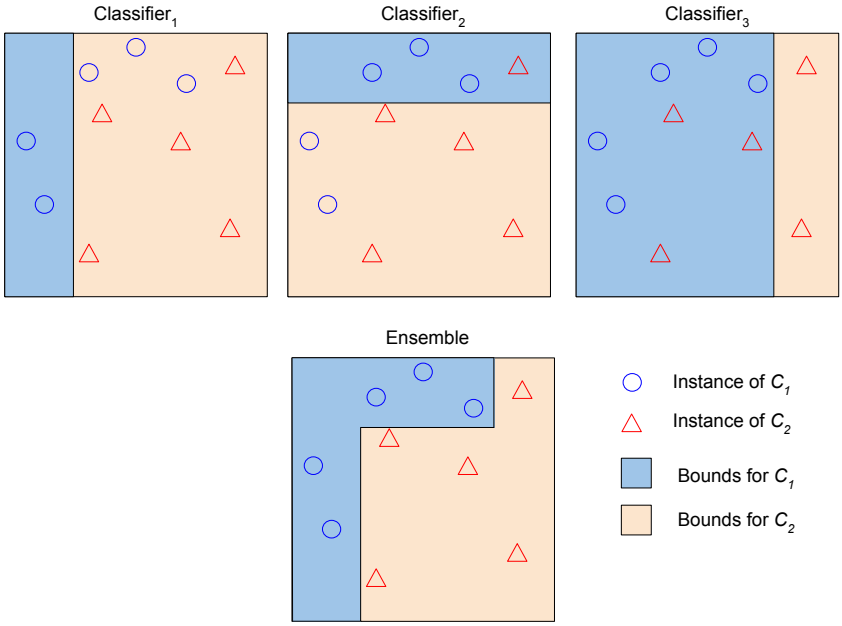

Figure 2.2: An ensemble classifier.

where a single classifier is unable to create a clean boundary between classes and can only isolate part of C_1 or C_2. Combining the votes of three different classifiers trained using the same sample, the ensemble classifier can perfectly fit to the training data.

The following sections present the most widely used ensemble classifiers.

2.5.1 Bagging

Bagging or Bootstrap Aggregating [21] is the process of aggregating multiple models (classification trees, linear regression models...) and producing a final outcome by averaging the outcome of the individual models in the case of numerical outcomes or by voting when predicting classes.

Usually the same learning algorithm is used to create the multiple models, albeit built with different training samples. These different samples will be bootstrap samples, thus the name: Bootstrap Aggregating. Bootstrap samples are of the same size as the original samples and they are created by randomly selecting instances of the original sample, with replacement, meaning the same instance can be selected more than once for the same sample, and sometimes not even once. Although not common, it is also possible to use different learning algorithms. Figure 2.3 shows a basic Bagging structure. From each bootstrap sample an individual model is built independently of the rest of models. During the testing phase, the class of each unseen instance is predicted by all individual

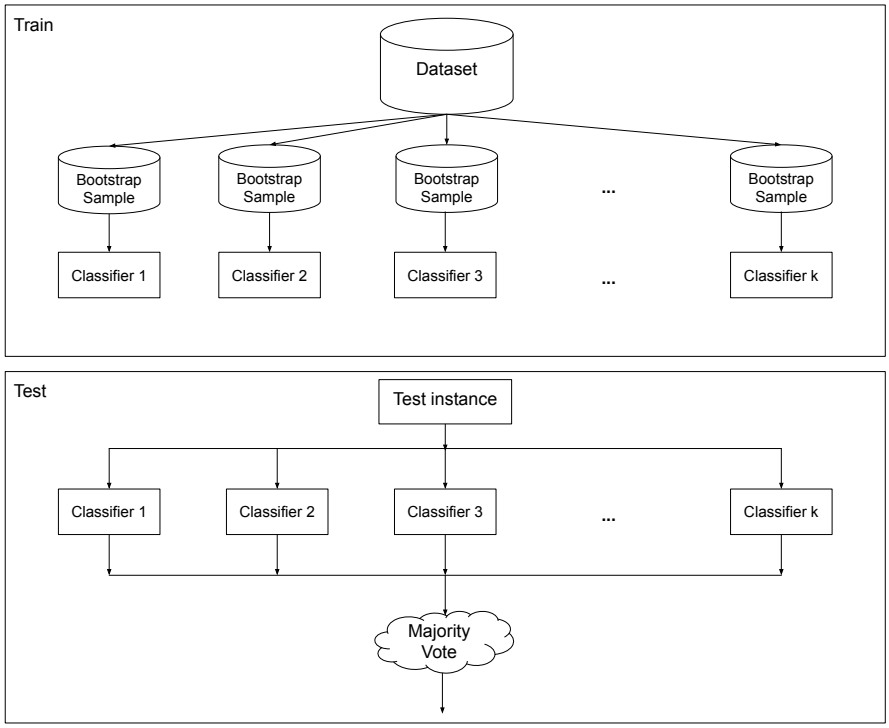

Figure 2.3: Training and test phases of Bagging.

models. The class voted by the majority is assigned to the instance. The weight of each vote from each individual classifier is inversely proportional to its error rate during the training phase.

This procedure offers better accuracy and more stability against changes in the training sample. The greater the instability of the base algorithm, the better the improvement. Tests run by Breiman show misclassification rates reducing in the ranges from 20% to 47% [21].

Because of these significant improvement, different variants of Bagging have been created since. Sub-Bagging [23], for example, uses subsamples instead of bootstrap samples.

2.5.2 Boosting

Boosting [135] aims to answer a question posed by Kearns [99] asking if a set of weak learning algorithms can create a stronger learning algorithm.

Boosting creates a series of classifiers sequentially. All instances start with the same weight, but from one iteration to the next, the training sample is reweighted. Samples misclassified by the last iteration gain weight for the next. In case the base algorithm cannot work with instance weights it is also be possible to resample the training sample where misclassified examples have a greater probability of being in the resampled data.

The final decision is made with weighted voting of individual classifiers where the weight of each classifier depends on its misclassification rate: the lower is the error, the bigger is the weight. Figure 2.4 shows a basic example of a Boosting algorithm.

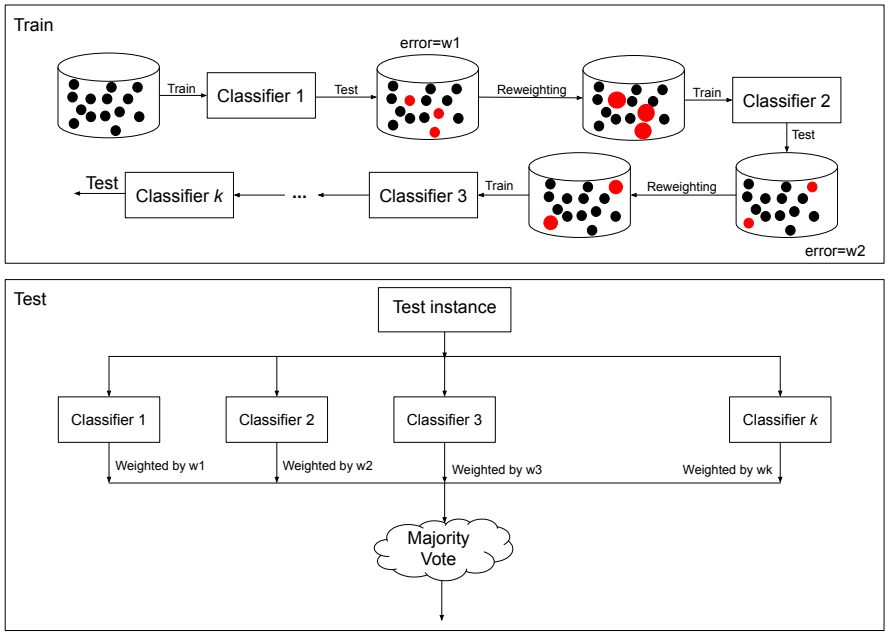

Figure 2.4: Taining and test phases of Boosting.

2.5.3 Random Subspace Method

Another type of ensemble algorithm are the Random Subspace Methods. They were initially proposed as the Random Forests algorithm [81] that creates an ensemble of decision trees. These methods work in a way similar to Bagging, but instead of creating new samples by picking instances at random, the new samples are created by picking features at random without replacement, so that each new sample has less features than the original sample.

From each one of these samples, a classifier is built. In the test phase, the class probabilities predicted by each individual model are combined to get the average class distributions, and the class with the greatest probability is assigned to the unseen instance. Figure 2.5 shows how these ensembles work.

2.5.4 CMM

CMM or Combined Multiple Models [43] is a meta-learner that tries to harness the accuracy gains of ensemble classifiers while retaining the explaining capability of single classifiers.

CMM uses Bagging as the ensemble classifier methodology. First the individual classifier is trained several times using bootstrap samples created from the original sample. Then, a new set of examples is randomly generated and classified according to the ensemble model. A new training sample is created

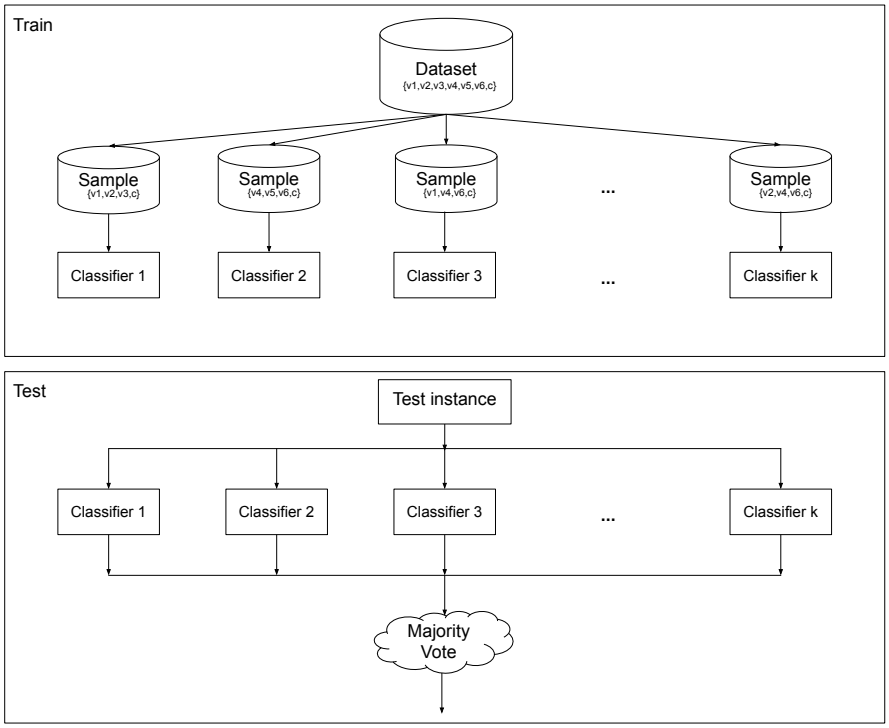

Figure 2.5: Training and test phases of a Random Subspace Methods.

combining the original training sample and the new generated examples. Finally, the learning algorithm is trained using this new sample and the obtained model used as the final model.

Using C4.5rules as the individual classifier, CMM retains on average 60% of the accuracy gains obtained from applying Bagging to C4.5rules while still generating a single, comprehensible, and more stable model [43].

CMM follows the *explanation by simplification* post hoc approach explained in Section 2.2.

2.6 Consolidation

The consolidation of decision tree algorithms is an approach that uses the ensemble voting process during the base classifier algorithm's building process to create a single classifier that keeps the explaining capacity of the base classifier algorithm [114, 121, 122].

The first consolidated algorithm was C4.5, which resulted in the Consolidated Tree Construction (CTC) algorithm. CTC was originally conceived to tackle a car insurance claim fraud problem where class imbalance was present,

and where the comprehensibility of the models was required.

Consolidation works by first creating multiple samples like other ensembles do. Bagging, for example, creates bootstrap samples: samples with the same size as the original but selecting examples using replacement. CTC can use different types of samples, but originally used bootstrap and stratified samples: subsamples that keep the original dataset's class distribution. Stratified samples had a size relative to the original dataset.

For understandability, CTC could be regarded as a Bagging of C4.5 decision trees that are built concurrently, not independently as in Bagging. From each subsample, a C4.5 (sub)tree begins to grow, but instead of splitting the subset according to its own split, each subtree casts a vote of the best split for its particular sample, but all subtrees comply to the majority vote and split their samples accordingly, even if it is not what they originally voted for. If the most voted variable is a discrete variable, a child is created for each possible value, like in the C4.5 algorithm. If the most voted variable is continuous, the algorithm collects all of the proposed split points and selects the median value among them. Originally, the average value was considered, but results showed that the median yielded better results. The process continues until the majority decides not to split anymore. When a consolidated tree has to classify new examples, the average of class membership probabilities of that leaf node on each subtree are assigned and the class with greatest probability is assigned to the instance like in the C4.5 algorithm. Figure 2.6 shows this process graphically and Algorithm 2.5 displays the pseudocode.

The visual example shows that from the original dataset several samples are created and from each sample a C4.5 tree begins to be built. At the root node two of the shown C4.5 trees vote to split the node using the discrete *gender* variable. All trees split their sample by that variable, even tough, for example the second tree proposes splitting by the *age* variable using 33 as the threshold. At that point the root node is consolidated. In the second node two trees propose cutting by the age variable but they propose different cut points (28 and 30). The median among the proposed should be used. In this case there are two possible values and the median should be the average of the two, but since C4.5 uses actual variable values for the split, the median's immediately higher value (30) is used as the consolidated cut point. This continues until the majority of trees vote not to split anymore.

In essence, a CTC classifier could be seen as a Bagging of C4.5, without the restriction of limiting to bootstrap samples, where all of the individual decision trees share the same structure and node conditions, with each subtree having different class probabilities on their leaves. From an implementation point of view, however, it is not necessary to actually build all of the subtrees. Since all of the subtrees have the same structure, the actual output model of the algorithm is a decision tree with the subtree's identical structure, and leaf class probabilities computed from the probabilities of the subtrees. Thus, as the base algorithm consolidation is applied to (C4.5) is comprehensible, CTC is also comprehensible, even if it combines the knowledge of multiple samples like full-fledged black-box ensembles do.

Input:
S: training data set
***N_S*:** number of samples
R_M: resampling method
Function `consolidate`(*S, N_S, R_M*)**:**
  resample S into $NextS = \{S_1, S_2, S_3, \ldots, S_{N\_S}\}$ using R_M
  `expand_consolidated`(*NextS, N_S*)
  return
Function `expand_consolidated`(*S,N_S*)**:**
  for *i=1 to N_S* **do**
    find best split $(V,t)^i$ with variable V and threshold t based on S_i
  end
  Compute consolidated split (V_c, t_c) based on all $(V,t)^i$
  if (V_c, t_c) == *No Split* **then**
    turn node into leaf
    consolidate label based on S_i where $i \leq 1 \leq N_\S$
    return
  end
  for *i=1 to N_S* **do**
    split S_i into $S_i^1, S_i^2, \ldots, S_i^m$ based on (V_c, t_c) where m is determined by the values V_c can take
  end
  for *j=1 to m* **do**
    NextS = $\{S_1^j, S_2^j, \ldots, S_{N\_S}^j\}$
    `expand_consolidated`(*NextS,N_S*)
  end
  return

Algorithm 2.5: Consolidation process.

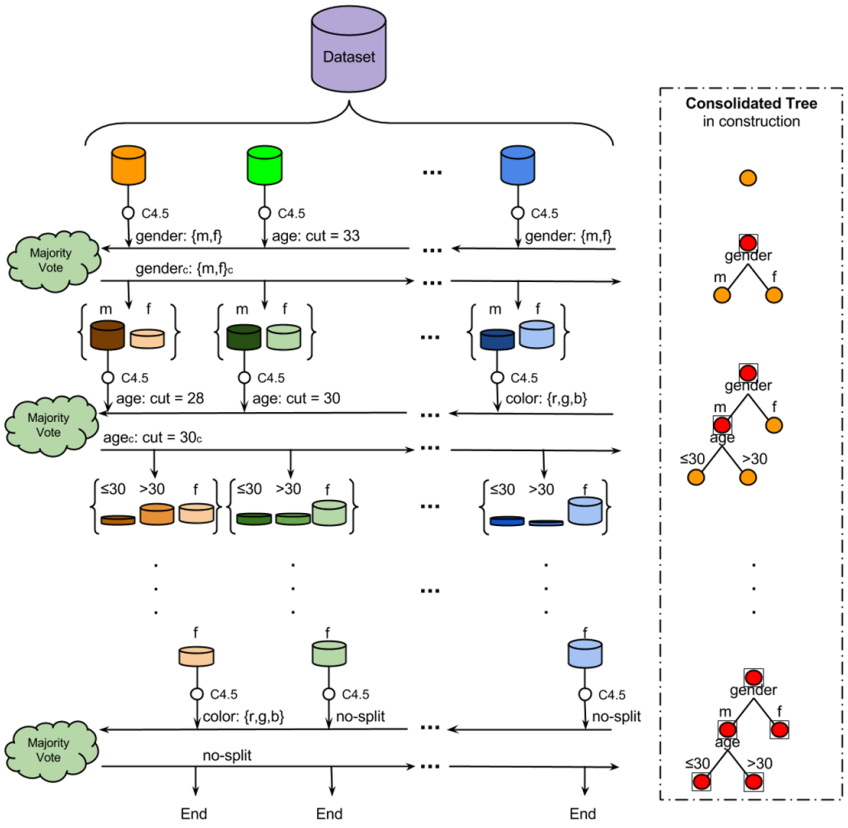

Figure 2.6: Visual representation of the CTC algorithm's construction process.

Abbasian, Drummond, Japkowicz, and Matwin [1] coined the term "Inner-Ensembles" to encompass approaches such as this, where the ensemble process is applied in the training phase instead of the testing, or classification, phase. In [1] they applied the procedure to the K-means clustering algorithm and to the network-building phase of the Bayesian Network classification algorithm. They also mentioned consolidation, which had been presented years prior, as en example of Inner Ensemble.

2.6.1 Previous Research on Consolidation

Since was first presented [117] there has been plenty of research on consolidation. The initial studies revolved around comparing it with CTC's base classifier, C4,5. Using the error rate as the performance measure over standard datasets, although using enough samples (>10) CTC always got better average results than C4.5, initially, statistically significant differences were only found for some datasets, not globally. Measuring the similarity of trees trained with different samples, however, CTC always shows better stability, in some cases even reaching full convergence of consolidated trees built using different sets of samples [117, 116, 118, 121, 120]. The differences in the stability between CTC and C4.5 are statistically significant. The complexity of the models, calculated counting the amount of internal nodes of a tree, is smaller in the case of CTC, making consolidated trees easier to understand. This is important because as Turney and Domingos separately pointed out "the engineers are disturbed when different batches of data from the same process result in radically different decision trees. The engineers lose confidence in the decision trees, even when we can demonstrate that the trees have high predictive accuracy." [146] and "a single decision tree can easily be understood by a human as long as it is not too large" [43].

Once the benefits of CTC over C4.5 were clear, continuing research focused on comparing CTC with algorithms that work in a similar manner, specifically Bagging and CMM. Bagging does not offer a comprehensible explanation since its decision is a combination of the votes of several classifiers. Although Bagging obtains, on average, better results than CTC (without statistically significant differences) [115], CTC has a better discriminating ability than CMM, with statistically significant differences regarding the stability of trees built with similar subsamples and creating simpler, more understandable models [74, 115].

Another aspect that has been a main focus on research about consolidation is how samples are created. Consolidation requires resampling, which makes CTC ideal for problems where the resampling of the original sample is necessary, such as the class imbalance problem. A comparison between CTC and C4.5 using subsampling achieved a statistically significant improvement on the error rate in favor CTC while maintaining stability and keeping the explaining capacity [73]. As explained later in Section 2.8.1.1, the class imbalance problem requires the use of better suited performance metrics such as the Area Under the ROC Curve (AUC, explained later in Section 2.8.1.1). Comparing the AUC measure of CTC and C4.5 over imbalanced datasets, CTC achieves better average results. In

general, the use of intelligent resampling methods such as SMOTE is beneficial for CTC [7]. While obtaining better results than C4.5 and SMOTE combined [6], CTC's results are improved even more with the use of intelligent resampling methods. For the analyzed datasets, in general CTC works best with class distributions close to the balanced distribution (40-60%). Because of this CTC can benefit from the modification of the original class distribution of the problem [122].

Several resampling strategies have been tested, and results have been dissimilar depending on the datasets. However, from a global point of view, stratified subsamples with a size of 75% of the training sample's size proved to obtain the best results, with 10 subsamples being enough to beat the error rate of C4.5 while achieving a lower complexity. On average the accuracy of CTC improves as the number of samples used increases, reaching a stability in accuracy and tree structure from 50-70 subsamples onward. Using bootstrap samples only benefits CTC in some cases [73], so the use of bootstrap samples is not recommended for CTC. The optimal class distribution for CTC does not always coincide with the optimal class distribution for C4.5, its base classifier [5].

The latest study regarding CTC [12] prior to this thesis, tested the algorithm with highly imbalanced datasets, extracting balanced datasets with a subsample size proportional to the amount of minority class examples. This study defined a first notion of *coverage* in order to determine the number of subsamples to use for each dataset and subsample size instead of fixed subsample numbers as it was done previously. However that notion differs from the one proposed in this dissertation on the fact that it does not ensure the representation of examples of all classes equally. This study shed some promising results, with CTC ranking second against 22 evolutionary and non-evolutionary classic algorithms in a context of 33 imbalanced datasets.

2.7 The PART ruleset algorithm

The PART ruleset induction algorithm combines the two most-common approaches for rule induction: extracting rules from decision trees and the separate-and-conquer approach. It was proposed by Eibe and Witten in "Generating Accurate Rule Sets Without Global Optimization" [55].

C4.5rules [128] was proposed along with the C4.5 decision tree algorithm and follows the first approach by transforming an unpruned tree into a ruleset by creating a rule for each leaf with the decisions between the tree's root node and the leaf. Then each rule is simplified by greedily removing conditions to minimize the rule's estimated error. Then, a subset of rules is selected using a criterion based on the minimum description length [131]. Finally, rules are removed from the ruleset as long as the removal reduces the error rate on training data. However, this optimization process is lengthy and complex.

Separate-and-conquer approaches create a rule, remove the examples covered by the rule, and repeat the process until the whole training set is covered. Fürkranz [61] showed that it is most effective to prune each rule immediately

after creating it and using a separate stopping criteria to decide when to stop creating new rules. RIPPER follows the separate-and-conquer strategy by ordering classes from least to most populous in the training sample, leaving the most-populous class as the default class, and creating rulesets for each class sequentially. Pruning a rule immediately after creating it can lead to a very aggressive pruning.

Both approaches (C4.5rules' and RIPPER's) follow a two-step process: first an initial model is created and then a heuristic improves its discriminating capacity. Even though this post-induction process generally increases the accuracy and decreases the complexity of the model, the process is rather complex and heuristic.

PART "avoids global optimization but nevertheless produces accurate, compact rule sets" [55]. This lack of global optimization aims to avoid the issues of C4.5rules and RIPPER. It adopts the separate-and-conquer strategy by building a partial decision tree, extracting a rule, removing the instances covered by this rule, and repeating the process until the entire training sample is covered.

Unlike C4.5rules, PART only uses a portion of the tree to create the rule. The idea of repeatedly creating decision trees only to use part of it is not far fetched. Using a pruned tree to extract a rule instead of incrementally adding conditions to a rule avoids the overfitting of pure separate-and-conquer strategies. Combining separate-and-conquer strategies with decision trees adds flexibility and speed.

The waste of time generated by repeatedly creating decision trees only to use part of it can be accelerated by using partially developed trees instead of fully explored trees. Partial decision trees do not expand the tree in the same order as the original algorithm does. First, the root node is expanded as the decision tree algorithm would. PART uses C4.5 for the decision trees so the node is expanded using the variable with the highest gain ratio, and instances (even those with missing values) are divided into subsets just like C4.5 does. Among the newly created child nodes, the one with the lowest entropy is selected to be expanded next. According to the authors, this is because nodes with lower entropy values are more likely to produce smaller subtrees and, thus, a shorter and more general rule. If a node cannot be expanded its sibling nodes are considered. If at some point all nodes of a subtree have been treated and left as leaves pruning is attempted. Only half of C4.5's pruning process is applied (subtree raising). If the subtree is not pruned, the partial tree construction process stops. If the subtree replacement does happen, then the process continues by analyzing the newly replaced node's untreated siblings. If no replacement fails the final decision tree will be a single root node covering the entire (remaining) sample.

Algorithm 2.6 shows PART's main procedure. This procedure uses the `expand_PART` function from Algorithm 2.7 and the function `get_rule` which first identified the biggest treated leaf, and then discards any tree branch not leading to said leaf node. Algorithm 2.7 shows the partial tree construction procedure. The functions `new test, best, gain_ratio` and `average_gain_ratio` found in that algorithm work the same as they did in Algorithm 2.2. The `sort_by_entropy` function sorts the subsets created from the split by their en-

tropy, from lowest to greatest. Figure 2.7 shows an example of a partial decision tree being built, and a rule is extracted. The following steps happen in the figure:

1. In this step the root node has been split into three nodes. These three nodes (2, 3, 4) are untreated for now.

2. The node with the lowest entropy (Node 3) has been expanded further into two nodes. Node 3 is now treated.

3. In this step two nodes have been treated. Node 6 had an entropy of 0, meaning all instances were of the same class and cannot be split further. Next Node 5 has been treated, and it cannot be split further.

4. All children of Node 3 have been treated so a subtree replacement is attempted and it succeeds. Node 5 and Node 6 are pruned and Node 3 becomes a leaf, and the algorithm backtracks.

5. Back at the same level as Node 3, the next untreated node with the lowest entropy is treated (Node 4).

6. In this step both Node 7 and Node 8 are treated. Node 7 is already class-homogeneous and cannot be split. Node 8 has examples of different classes but cannot be split either.

7. In this case the replacement of Node 4 fails and the partial process is stopped. Node 2 is left untreated. Among the treated leaves (3, 7 and, 8) the one with the biggest size (Node 3) is chosen to extract the rule.

Once the partial tree is built the leaf covering most examples is used to extract a rule. This rule contains the decisions between the root node and the selected leaf. This is the most general rule. The author's experimented using the most accurate rule but this did not increase the ruleset's accuracy. Instances covered by this rule are removed from the training sample and if there are still uncovered instances the process is repeated.

The authors claim that this procedure ensures that the aggressive pruning issues found in RIPPER do not happen.

When using the ruleset to classify new instances, the instance is tested using all rules. Each rule will return a positive weight (if the rule covers the instance) and the class probabilities according to that rule. The final classification is achieved by combining the outcome of each rule, the same way C4.5 does.

An empirical study by PART's authors compared the algorithm to C4.5, C5.0 and RIPPER, and concluded that the main advantage of PART is not performance but simplicity. Even if the results of that study could not conclude that PART performed significantly better (or worse) than any of the other algorithms, PART has become a widely used rulesets algorithm. At the time of writing it has been cited over 1200 times according to Google Scholar.

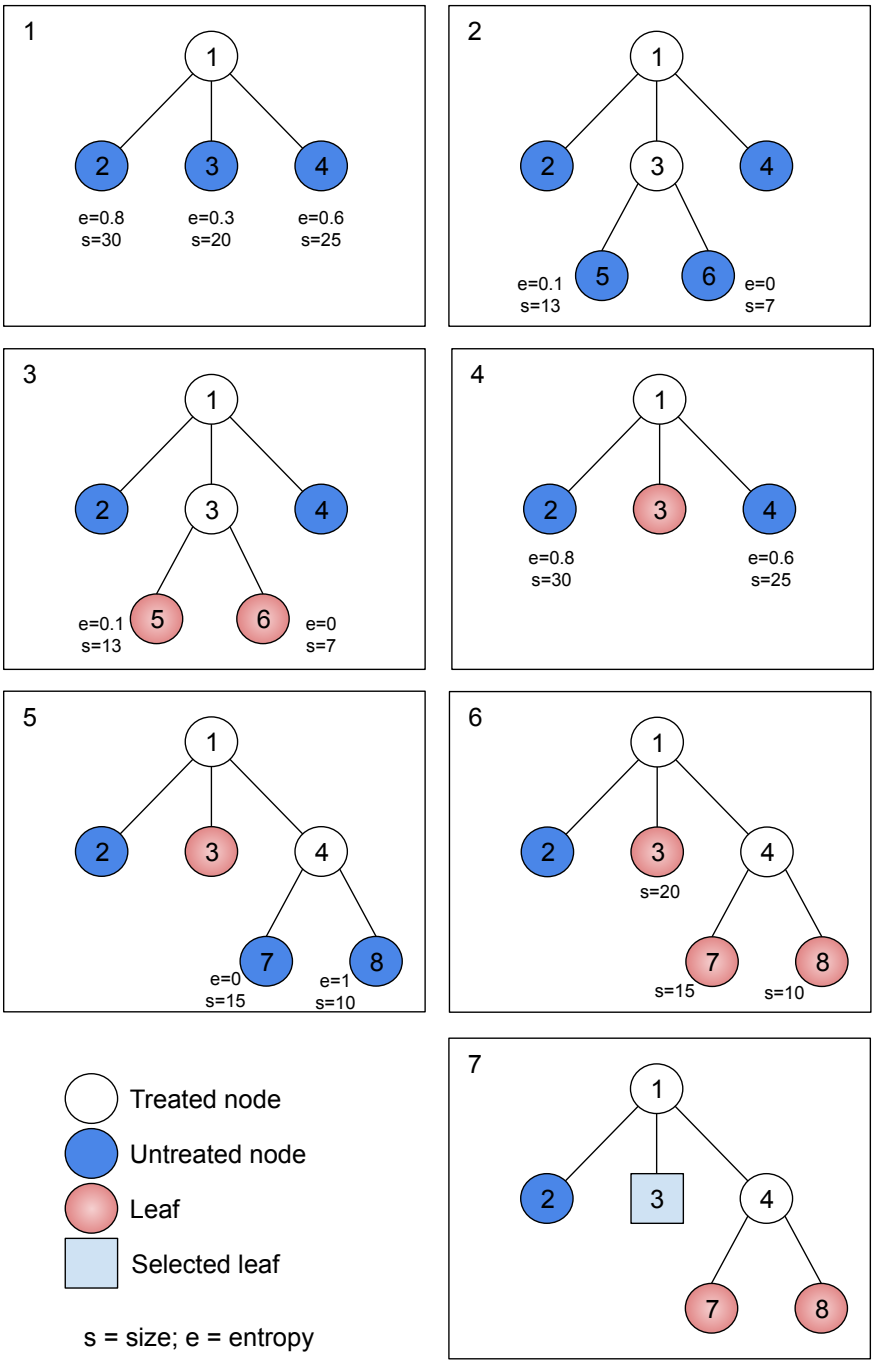

Figure 2.7: Example of the process of building a partial tree and extracting a rule.

Input:
S: training data set
Function `PART_ruleset(`S`):`

```
ruleset = ∅
while S ≠ ∅ do
    partial_tree = expand_PART(S)
    R = get_rule(partial_tree)
    foreach instance I in S do
        if R covers I then
            S = S \ I
        end
    end
    ruleset = ruleset ∪ R
end
```

Algorithm 2.6: PART's ruleset construction algorithm.

```
Input:
S: training data set
Function expand_PART(S):
    mark S as treated
    if S is empty then
        turn node into leaf
        assign S same label as parent
        return
    end
    if S meets stop criteria then
        turn node into leaf
        assign S label of most common label of instances in S
        return
    end
    if all cases on S are members of class C_j then
        turn node into leaf
        assign label C_j to S
        return
    end
    foreach V_i in V do
        if V_i is discrete then
            T_i = new_test(V_i)
        end
        if V_i is continuous then
            threshold = best_split(V_i)
            T_i = new_test(V_i, threshold)
        end
        T = T ∪ T_i
    end
    gr_avg = average_gain_ratio(T)
    T_b = best(T)
    if gain_ratio(T_b) < gr_avg then
        turn node into leaf
        assign S label of most common label of instances in S
        return
    end
    foreach instance I_j in S do
        foreach outcome O_i in T do
            if o_i == I_ji then
                S_i = S_i ∪ I_j
            end
        end
    end
    sort_by_entropy(S_1, S_2, S_3, ...)
    foreach S_i in S do
        expand_PART (S_i)
        attempt subtree replacement on S_1
        if replacement failed then
            return
        end
    end
    return
```

Algorithm 2.7: PART's partial C4.5 tree construction algorithm.

2.8 Evaluation of Supervised Classification Algorithms

With so many classification algorithms in existence, metrics have been developed to compare the performance of multiple classifiers. However, achieving a better average value for a metric or measure is not enough to prove that the performance of a classifier is better than another. The differences in performance between classifiers need to be tested for statistical significance to ensure that the differences are not due to chance.

The following sections will first, lay out a set of metrics to evaluate and compare classifiers, and a set of statistical tests the significance of differences between two or multiple classifiers.

2.8.1 Metrics to Evaluate Classifiers

The most straightforward way to measure how effective classifiers are is to measure how well they classify examples unseen during the training stage. This is also called their ability to generalize or discriminate.

When the models created by the algorithm posses the ability to explain how a decision is made, their structure takes importance. Another way to compare explainable models is to measure the complexity of the models.

2.8.1.1 Metrics to Evaluate a Classifier's Discriminating Capacity

The discriminating capacity of a classifier is usually measured from the point of view of *accuracy*, i.e. the ability to correctly classify examples that were not used during training. This ability is also called *discriminating capacity* or *capacity of generalization*.

In a real world problem, when new instances are classified, it is the goal of the classifier to correctly assign a class to the instance, so the class is not known, and thus, the classifier's accuracy cannot be assessed. However, when an algorithm is created or multiple algorithms are compared, their accuracy is calculated by holding part of the training set apart. This *test set* is not used during the training phase, and during the testing set, the algorithms do not know or use the value of the class, but it is used to check whether the classifiers output is correct or not.

The most straightforward way of measuring how accurate a classifier is, is to measure the overall accuracy of the classification:

$$overall\ accuracy = \frac{correct\ classifications}{total\ classifications}$$

However, this is a measure that does not take any characteristic of the dataset into account, and in a lot of cases, is considered an inadequate measure. In a two-class problem where one of the classes has much fewer examples than the other (a problem with class imbalance), for example 10 instances of the minority class out of a total of 1000 instances, a model that simply classifies

any example as the majority class, regardless of its attribute values, achieves an overall accuracy of 99% which, at least at first glance, looks great, but has no real classification value, as the class of interest is usually the minority class, and this model fails completely at identifying it.

Most accuracy measures are based on what is called a *confusion matrix*. Such a matrix shows the relationships between the actual class of the instances, and the predictions made by the classifier. Table 2.3 shows a confusion matrix for a two-class problem. In two-class problems, usually one of the classes is the concept of interest (e.g. the 'sick' class in a 'healthy/sick' domain), and this class is referred to as the *positive class*, whereas the other class is called the *negative class*.

		actual class	
		+	-
classified as	+	TP [a]	FP [b]
	-	FN [c]	TN [d]

[a] True Positives
[b] False Positives
[c] False Negatives
[d] True Negatives

Table 2.3: A confusion matrix.

Using a confusion matrix, the above mentioned overall accuracy can be represented in the following way:

$$overall\ accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

When looking at class-specific measures, the simplest are the *true positive rate* (referred to as *TPrate* and also called *sensitivity* or *recall*), and its counterpart, the *true negative rate* (referred to *TNrate* and also called *specificity*). These rates measure how much of each class is classified correctly.

$$TPrate = \frac{TP}{total\ positive\ class} = \frac{TP}{TP + FN}$$

$$TNrate = \frac{TN}{total\ negative\ class} = \frac{TN}{TN + FP}.$$

These two measures are commonly combined by computing the geometric mean (GM) of the two:

$$GM = \sqrt{TPrate \times TNrate}$$

From the matrix above, similar metrics can be extracted for the incorrect class predictions:

$$FPrate = \frac{FP}{total\ negative\ class} = \frac{FP}{TP + FP}$$

$$FNrate = \frac{FN}{total\ positive\ class} = \frac{FN}{TN + FN}.$$

In the class imbalance problem given above, if a model classified all examples as majority class, no positive class examples would be classified correctly. This would mean that the value for TP would be zero, which in turn would result in a TPrate of zero, which in turn would result in a GM of zero. A problem with 99% overall accuracy would get a zero for another measure. This shows how measures that work class-by-class are much better suited. Specially in cases where class imbalance is present.

Similar to the recall, *precision* measures how many of the instances classified as positive, are actually positive:

$$precision = \frac{TP}{TP + FP}.$$

Combining both recall and precision, another metric that covers the entire confusion matrix can be derived. The F1-Score (also called the F-Score or F-Value).

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

Another interesting measure is Cohen's kappa statistic [35]. This metric determines how much the assessed classifier differs from a random or trivial classifier: that is, a classifier that simply bases its output on the apriori class distributions of the problem.

$$kappa = \frac{TP + TN - E(TP + TN)}{TP + TN + FP + FN - E(TP + TN)},$$

where $E(TP + TN)$ represents the chance of classifying correctly by random chance.

$$E(TP + TN) = \frac{(TP + FP)(TP + FN) + (TN + FN)(TN + FP)}{|S|^2}$$

where $|S|$ is the number of examples.

Finally, a widely used metric is the Area Under the Receiver-Operating Curve (AUC) [52]. The ROC curve can be represented in a two-axis chart and is used to represent the relationship between the TPrate and the FPrate.

On models that only the class label for unseen examples is given, this only allows for a single point. In this case, the curve starts from the (0,0) point, goes through the (TPrate,FPrate) point, and ends up in the (1,1) point. When the classifier, instead of assigning a single class to unseen instances, gives class membership probabilities as the output, each probability value in the model can be used as threshold to determine if an instance is positive (instead of the usual 0.5 threshold). The lower this threshold, more instances are classified as positive (more true and false positives). Using the TPrate and FPrates for each threshold, an actual curve can be drawn. The area below the ROC curve is used to assess the performance of classifiers. An ideal ROC curve would pass as close to the (0,1) point as possible to maximize the area under the curve. Figure 2.8 shows different ROC curves.

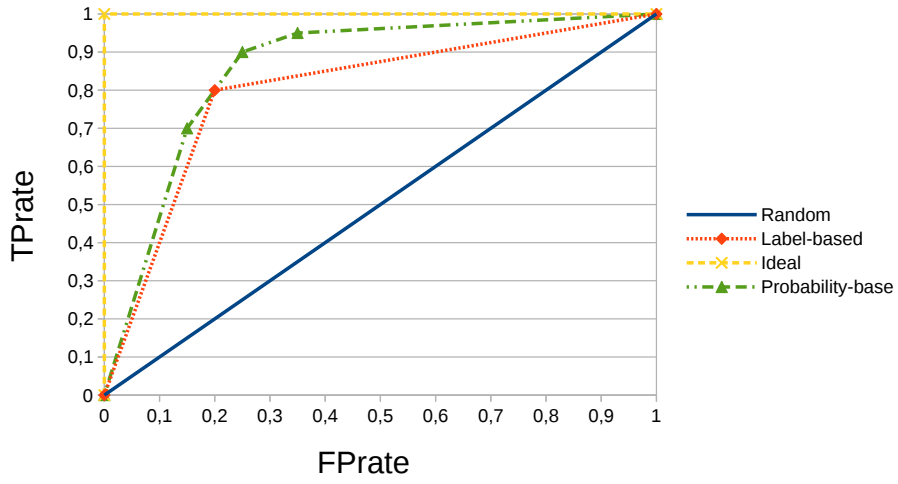

Figure 2.8: Multiple ROC curves displaying different scenarios.

AUC evaluates the classifier in multiple contexts of the classification space without assuming any misclassification costs or prior probabilities [137]. It is a robust metric [92] and it has been suggested that machine-learning algorithms should focus on optimizing their AUC rather than their overall accuracy [83].

When facing a multi-class problem, some measures have a specific implementation for this case. However, the most widely used approach is to compute k (where k is the number of classes) *one vs. all* values using a different class as base each time and averaging the results of all iterations weighted by the class' size:

$$Multi-class\ metric = \frac{\sum_{j=1}^{k} n_j \cdot metric(C_j, all\ classes \setminus C_j)}{n}$$

where n the size of the sample, n_j is the size of the j-th class (C_j) and k is the number of classes. C_j is the positive class of an iteration, and all other

classes combined make up for the negative class.

However, in this thesis, the following implementation has been used to compute the multi-class kappa statistic:

$$Multi - class\ kappa = \frac{n\sum_{i=1}^{k} h_{ii} - \sum_{j=1}^{k} T_{rj}T_{cj}}{n^2 - \sum_{j=1}^{k} T_{rj}T_{cj}}$$

where n is the number of examples, k is the number of classes, and T_{rj} and T_{cj} are row and column sums for the j-th class in the confusion matrix, and h_{ii} is the number of correctly classified examples for the i-th class.

2.8.1.2 Metrics to Evaluate a Classifier's Structural Complexity

When a classifier is comprehensible, there is another way how an algorithm can be evaluated: by measuring the structural complexity of the generated models.

If the comprehensibility of the classifier is desired, it is because a human operator will make use of the knowledge extracted by the classifier. The less complex, or in other words, the simpler a classifier it is, the easier it is for a human to be understood; so lower structural complexity equals greater comprehensibility. The classifiers accuracy should prevail over the simplicity of the models. However, with equally accurate classifiers, the Occam's Razor principle should be followed, and the simplest should be chosen.

It is not easy to select correct criteria to measure the complexity of a classifier [62]. However, in this dissertation, the only classifiers measured from this point of view are decision trees and rulesets (extracted from decision trees). This makes the choice of metrics much easier.

From a macro point of view, the easiest way of assessing the complexity of a ruleset is how many rules it contains. This can be translated into decision trees by how many leaf nodes it has; as the path from the root node to one of the leaves is the equivalent of a rule. In order to assess the real complexity of a ruleset, looking at the number of rules is not enough, as each rule could contain as few as one condition, but this number could grow to the dozens or hundreds in more complex rules. The number of rules is one of the metrics used in this thesis to measure the complexity of classifiers. Looking on a micro level, the complexity of the rules within a ruleset can be assessed by looking at the length of the rules, measured in the number of conditions they contain. Throughout this dissertation, this measure is referred to as *Length*. Length measures the average number of decisions in a rule (or the average number of nodes between the root of the tree and the leaves in a tree). This metric is tightly related to the decision support system nature of comprehensible classifiers. Most of the time, even in a massive ruleset or decision tree, an example will only be covered by a single rule or branch. Because of this, *Length* is an adequate way to measure how easy to use the classifier will be for a human user (e.g. How many conditions will a doctor have to check to diagnose a patient?).

These two metrics, *Number of Rules* and *Length* give a value to the two dimensions in which the classifiers built by ruleset and decision tree induction algorithms can be measured from the point of view of structural complexity.

2.8.2 Validation of Supervised Classification Systems

When the performance of a classifier has to be evaluated, the technique used to do so should be as "real" as possible. However, "real" test data is not labeled, as it is the classifier's job to assign that label. This means that in order to assess the discriminating capacity of a classifier, labeled data meant for training has to be used. Using part or the whole training set, the same one used to create the model, is not adequate because virtually any algorithm can create a complex enough model capable of correctly classifying the entire dataset. This accuracy might not, and probably will not, be achieved with examples not used in the training phase. Learning algorithms need to be able to generalize beyond the training set and be able to correctly identify previously unseen examples.

Usually part of the dataset is left out of the training process and used during testing to validate the performance. The easiest way to do this is to randomly exclude part of the training set. However, doing this once is not enough, because the classifier's good or bad performance could be attributed to chance when creating the partitions.

The most common way of doing this is to perform *cross-validation*. In cross validation the dataset is randomly divided into k folds. $k-1$ of the folds are used for training, and the last fold is used for testing. This process is repeated k times using a different fold for testing each time. This ensures that all instances are used for the test part. The results are obtained by averaging the performance of the k classifiers. Usually k is a small number like five or ten. However, this can be taken to the extreme, where the fold size is just one instance, so only one instance is tested each time, and the rest of the dataset is used for training. This is called *leave one out* and it has to be repeated for each instance in the training set, which makes it much more expensive computationally compared to regular cross validation. Figure 2.9 shows an example of cross-validation where the folds are stratified, meaning each fold keeps the same class distribution as the original dataset. In that Figure, the lighter shade of each box represents one class (majority class), and the darker shade the other class (minority class).

Usually cross-validation is repeated several times in order to reduce the effect of chance even more, with 5-run 5-fold cross-validation (5x5cv) and 10-run 10-fold cross validation (10x10cv) being very common examples.

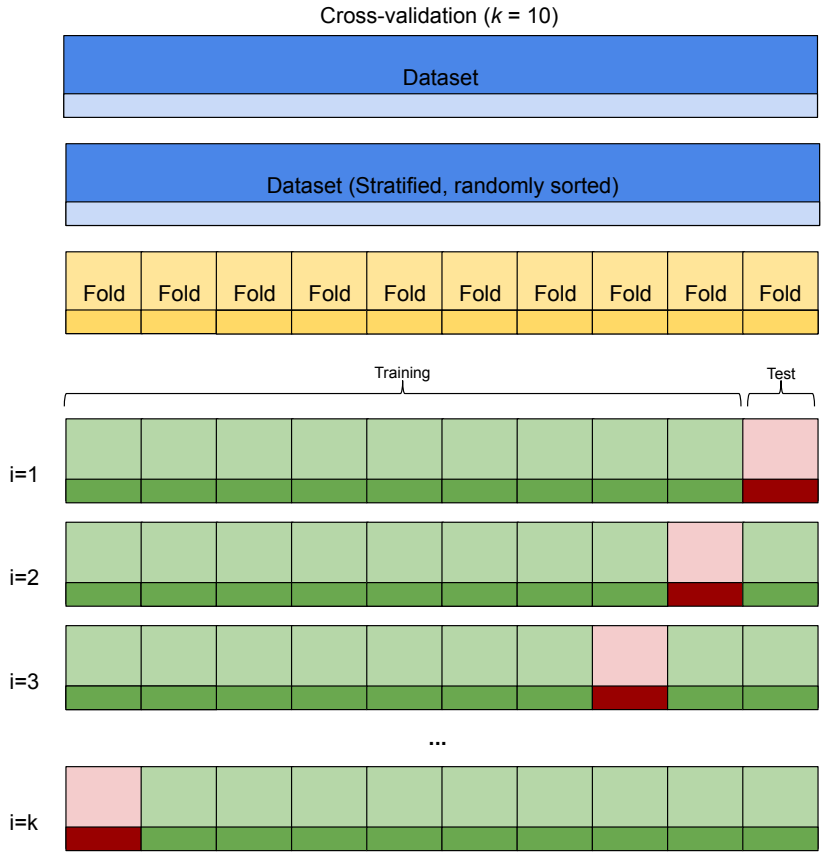

Figure 2.9: Example of cross validation for classification.

2.8.3 Tests for Statistical Significance

When a new machine learning system is proposed, a hypothesis is made that this new contribution yields an improvement over already existing competing systems. In recent years, the machine learning community has become aware of the need to statistically validate their results. In consequence, researches addressed the problem of comparing two classifiers on a single datasets. However, there is a more common situation, comparing multiple classifiers over multiple datasets. Until recently, there was no theoretical background on how the results of this kind of study should be tested for statistical significance, and researchers often resorted to methods tailored to the problem that either lacked a good statistical foundation or used the methods inappropriately. Demšar [39] analyzed the proceedings of the International Conferences on Machine Learning (ICML) between 1999 and 2003, and observed that many of the studies simply used an $n \times n$ matrix of McNemar's 5x2 cross-validation tests comparing all pairs of classifiers.

In [39], Demšar proposed multiple tests to validate the results of studies comparing classifiers over multiple datasets. These statistical tests compute the probability of the null-hypothesis being correct, where the null-hypothesis is that all compared algorithms perform equally and the observed differences can be attributed to chance. A clear distinction was made between systems to compare two classifiers, and systems to compare multiple (i.e. more than two) classifiers.

2.8.3.1 Tests to compare two classifiers

Looking at the papers of the International Conferences on Machine Learning, Demšar noted that some authors simply averaged the results of the classifiers over all datasets. However, as Webb stated [149], it is debatable whether the errors of different datasets are comparable and if averaging the errors of very different datasets is meaningful. Averages are very susceptible to outliers, which allows the extraordinary performance in a few datasets to completely skew the average value. For example, let's look at Table 2.4. This example shows values for two classifiers over four datasets, where classifier1 achieves the best results in three out of four datasets, but classifier2 achieves an exceptionally better performance in the last dataset. On average, classifier2 achieves a better performance but it only performs best in one out of four problems. It is possible that this behavior might be desirable in some situations, however it is usually preferred for an algorithm to perform best in most possible situations.

In order to compare two classifiers over multiple datasets, Demšar recommended using the *Wilcoxon signed-ranks test* [151], which is a non-parametric version of the paired t-test. Non-parametric tests make no assumptions about the probability distributions of the variables, in this case, the differences between classifiers.

This tests ranks the differences between the two classifiers for each dataset by their absolute value, without taking their sign into account. The biggest

dataset	classifier1	classifier2
d1	70	65
d2	75	70
d3	65	55
d4	75	100
average	71.25	72.5

Table 2.4: Example of outliers distorting the average value.

difference is given the highest rank, and the smallest difference is given the lowest rank, and the rest of differences are given the corresponding ranks. Then, the ranks of positive differences (where one classifier wins) are added on one side, and negative differences (where the other classifier wins) are added on the other side. If both algorithms perform equally for a dataset, half of the rank is added to each sum. If the difference between the sums is big enough, the differences in the performance of the classifiers are considered significant.

Demšar uses the following example. Consider the performance values (based on AUC) and differences shown in Table 2.5 and the ranks assigned to the differences. Let $diff_i$ be the difference of the i-th dataset, and rank($diff_i$) the rank assigned to the difference for that dataset. The ranks for positive differences (where classifier2 wins) are summed into R^+, and ranks for negative differences (where classifier1 wins) are summed into R^-.

$$R^+ = \sum_{diff_i>0} rank(diff_i) + \frac{1}{2}\sum_{diff_i=0} rank(diff_i)$$

$$R^- = \sum_{diff_i<0} rank(diff_i) + \frac{1}{2}\sum_{diff_i=0} rank(diff_i)$$

Let T be the smallest between R^+ and R^-. The test statistic z is computed in the following way,

$$z = \frac{T - \frac{1}{4}n(n+1)}{\sqrt{\frac{1}{24}n(n+1)(2n+1)}}$$

where n is the number of datasets and z is distributed normally for large numbers of datasets. For smaller numbers of datasets the critical value of T is fixed and can be found in the literature [39].

If the assumptions made by the paired t-test are met, it is more powerful than the Wilcoxon Signed ranks test. More powerful tests are able to find significant differences not found by less powerful tests. However, when comparing two classifiers over datasets there is no way to assure these assumptions are met. The Wilcoxon Signed ranks test is also very robust against outliers, as the

dataset	classifier1	classifier2	difference	rank
d1	.763	.768	.005	3
d2	.599	.591	-.008	7
d3	.954	.971	.017	9
d4	.628	.661	.033	12
d5	.882	.888	.006	5
d6	.936	.931	-.005	4
d7	.661	.668	.007	6
d8	.583	.583	.000	1
d9	.775	.838	.063	14
d10	1.000	1.000	.000	2
d11	.940	.962	.022	11
d12	.619	.666	.047	13
d13	.972	.981	.009	8
d14	.957	.978	.021	10

Table 2.5: Example differences between two classifiers.

most exceptional performances only sum a little more than than the next best performance.

A final way to compare two algorithms is counting the number of wins, losses and ties for each algorithm, and using the *sign test*, where, simply put, a dataset performs significantly better if its number of wins is greater than $N/2 + \sqrt{N}$. Even though this test does not make any of the assumptions the t-test makes, it is much less powerful than the Wilcoxon signed-ranks test.

In summary, Demšar recommended the use of the Wilcoxon signed-ranks test when comparing two classifiers over multiple datasets. This is the test used throughout this thesis when such comparisons are made.

2.8.3.2 Tests to compare more than two classifiers

Even though the tests proposed to compare two classifiers were not designed to compare multiple classifiers, in the past a lot of authors used them for such purpose, usually by creating a matrix of pairwise comparisons comparing all possible pairs and then extracting conclusions from this matrix. In the formulas in this section, n denotes the number of datasets, while in this section k refers to the number of compared classifiers.

A test suitable to compare the differences between more than two systems over the same datasets, is the repeated measures ANOVA test [54]. Unfortunately, the same way the paired t-test does for two classifier comparisons, the ANOVA test also makes assumptions. ANOVA assumes that the differences

between algorithms follow a normal distribution, and also sphericity, where it is required for the differences to have an equal variance. When comparing machine learning algorithms it is not possible to assure the differences meet these assumptions.

The same way the Wilcoxon signed-ranks test is the non-parametric version of the t-test, the Friedman test [58, 59] is the non-parametric version of the repeated-measures ANOVA test.

The Friedman test ranks the algorithm for each dataset, so that the algorithm performing best gets a rank of 1 and the algorithm performing worst gets a rank of k, where k is the number of compared algorithms. The average ranks over all datasets are used to test whether differences exist among all compared algorithms.

Let r_i^j be the rank of the j-th algorithm over the i-th dataset, so that the average rank R_j is computed $R_j = \frac{1}{n}\sum_i r_i^j$. Using these average ranks the χ_F^2 statistic is computed the following way.

$$\chi_F^2 = \frac{12n}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right].$$

The χ_F^2 test statistic is distributed according to the chi-squared distribution with $k-1$ degrees of freedom when the number of datasets and algorithms are big enough. For smaller k and n numbers the critical numbers can be found on [158, 140].

Iman and Davenport [88] later proposed an extension after showing the Friedman statistic is too conservative. Their F_F statistic follows the F-distribution with $k-1$ and $(k-1)(n-1)$ degrees of freedom and is computed the following way.

$$F_F = \frac{(n-1)\chi_F^2}{n(k-1)-\chi_F^2}$$

Demšar proposed the Iman-Davenport extension of the Friedman tests to compare multiple classifiers. Later, García et al. [64] proposed a set of advanced non-parametric tests that can be used in the same situations where the Friedman test can be used, and specially recommended these tests in specific situations.

One such test is the Friedman Aligned ranks test. This test is very similar to the original Friedman test, but instead of creating independent ranks for each dataset, a global ranking is computed, ranking all classifier-dataset combinations from 1 to kn. In order to do this, differences are calculated between the performance of an algorithm for a dataset and the average of all classifiers for that dataset. The differences between the performance for a classifier for a dataset, and the average of all classifiers for that particular dataset are called aligned observations, and the ranks assigned to these observations are called aligned ranks. Table 2.6 shows a simple example with four classifiers and five datasets showing how Friedman Aligned ranks are calculated.

	c1	c2	c3	c4	average	diff1	diff2	diff3	diff4	ranks1	ranks2	ranks3	ranks4
d1	.770	.775	.720	.755	.755	.015	.019	-.035	.000	6.5	2.5	19	12
d2	.850	.811	.855	.843	.840	.011	-.029	.015	.003	8	17	6.5	11
d3	.968	.970	.957	.915	.953	.016	.017	.004	-.037	5	4	10	20
d4	.911	.915	.937	.910	.918	-.008	-.003	.019	-.009	14	13	2.5	15
d5	.595	.521	.560	.544	.555	.040	-.034	.005	-.011	1	18	9	16
Avg. Ranks										8.9	10.15	12.1	14.8

Table 2.6: Example of Friedman Aligned ranks.

The test statistic for this test can be computed the following way, where $\hat{R}_i^2$ is the square of the sum of ranks for the i-th dataset and $\hat{R}_j^2$ is the square of the sum of ranks for the j-th classifier.

$$T = \frac{(k-1)[\sum_{j=1}^{k} \hat{R}_j^2 - (kn^2/4)(kn+1)^2]}{\{[kn(kn+1)(2kn+1)]/6\} - (1/k)\sum_{i=1}^{n} \hat{R}_i^2}.$$

The Friedman Aligned ranks test can be used to replace the Friedman test in any situation where the Friedman test is valid, and it is specially recommended when the number of compared algorithms is low (less than five). The original Friedman test does not allow inter-dataset comparisons, only intra-dataset ones. However, when the number of compared algorithms is low this may be a disadvantage, and the use of inter-dataset comparisons is desirable.

Some of the experiments in this thesis compare only four algorithms. As the Friedman Aligned ranks is recommended over the Friedman test in this situation, and the Aligned ranks test can also be used instead of the original Friedman test in the rest of multiple classifier comparisons, the Friedman Aligned ranks test has been used throughout this thesis when comparing multiple algorithms.

If the Friedman Aligned ranks test determines that there are significant differences between the compared algorithms, a post hoc test has to be performed in order to find out between which pairs of algorithms differences exist.

Demšar initially proposed the Nemenyi [109] and Bonferroni-Dunn [49] tests, recommending the Nemenyi test to compare all algorithms against each other and the Bonferroni-Dunn test only when comparing one control algorithm (a new proposal, for example) against the rest of competitors, as the power of Bonferroni-Dunn test is greater on $1 \times n$ tests than $n \times n$ tests. The benefit of these two tests lies in that they are one-step tests and they use fixed critical distances (CD) based on the number of competitors, so the significant differences can be graphically represented in a very simple manner using CD-diagrams as proposed by Demšar [39]. However, these tests are not considered adequate anymore.

Other procedures adjust the significance level (α) dynamically and are classified into step-up, step-down, and two-step procedures. These procedures first require to compute the p-value of all pairwise comparisons to be made. When the Friedman Aligned ranks test is used to determine the differences, the test statistic can be computed in the following way

$$z = (\hat{R}_i - \hat{R}_j)/\sqrt{\frac{k(n+1)}{6}}$$

where $\hat{R}_i$ and $\hat{R}_j$ are the average Friedman Aligned ranks achieved by the i-th and j-th algorithm.

Demšar proposed, and later Derrac et. al. [40] confirmed, that the Holm post hoc procedure [82] should be used for $1 \times n$ tests due to its simplicity and high power. For $n \times n$ tests, Derrac et. al. determined the Bergmann-Hommel [19] test to be the best performing. These are the two post hoc tests used throughout the thesis.

The Holm procedure is a step-down test. This procedure first orders the p-values for all pairwise comparisons with most significant (smallest) values first. Each p-value related to a pairwise *null* hypothesis. Holm's procedure tests hypotheses sequentially. If one hypothesis is rejected (meaning it cannot be determined that classifier performance is unequal), the next one is tested. But if a hypothesis cannot be rejected when $p_i > \alpha/(k-i)$, the remaining hypotheses (those with a greater computed p-value) are retained.

The Bergmann-Hommel procedure is a very computer-intensive method. This method first creates a set of accepted hypotheses consisting of all pairwise comparisons where $p > \alpha/k$. If the entire set of hypotheses is exhaustive (i.e. all hypotheses could be true), all exhaustive subsets of hypotheses will be computed (hence the high computational cost), and the subsets not present in the accepted hypothesis set are rejected.

Finally, García and Herrera [65] noted that when multiple pairwise comparisons are performed, a p-value reflects the probability error of one comparison, but it does not take into account the remaining comparisons of the same family. They suggested adjusting the p-values accordingly instead of dynamically changing the α, as having a single significance threshold allows to compare the probabilities of different hypothesis tests. For the Holm test the adjusted p-value (APV) is computed like $APV_i = min\{v; i\}$, where $v = max\{(k-j)p_j : 1 \leq j \leq i\}$ and k is the number of hypotheses. In order to compute the APV for the Bergmann-Hommel procedure, Algorithm 2.8 has to be followed.

2.9 Previously developed software

In this thesis, several pieces of software developed by other researchers have been used.

Every new contribution of this thesis has been developed using the research group's *Haritza* (Oak in Basque) platform. This platform is dedicated to decision tree and decision tree based algorithms. The original CTC algorithm and the C4.5 implementation used in this theses also belong to that platform.

The statistical tests for significance used in Chapter 3 and Chapter 5 have been using the *Non-Parametric Statistical Analysis* module of the *KEEL* tool [9, 8].

The results of the statistical tests are displayed in figures found on Section 4 such as Figure 4.1 created using the R package *scmamp* [25].

Finally, public implementations for the contributions of this thesis have been made available for the WEKA platform.[75]

2.10 Work used as reference for the experiments in the thesis

This section describes an article heavily referenced throughout the thesis. This work, titled '*Genetics-Based Machine Learning for Rule Induction: State*

Input:
p: p-values for all pairwise hypotheses
k: number of compared classifiers
Function `bergman_hommel(`p,k`):`

```
for all i do
    APV_i = p_i
end
foreach j = k-1,k-2,...,2 do
    B = ∅
    foreach i, where i > (k-1-j) do
        c_i = (j · p_i)/(j + i − k + 1)
        B = B ∪ c_i
    end
    c_min = min(B)
    if APV_i < c_min then
        APV_i = c_min
    end
    foreach i, where i ≤ (k-1-j) do
        c_i = min(c_min, j · p_i)
        if APV_i < c_i then
            APV_i = c_i
        end
    end
end
```

Algorithm 2.8: APV computation for the Bergman-Hommel approach.

of the Art, Taxonomy, and Comparative Study' by Fernández, García, Luengo, Bernadó-Mansilla, and Herrera [53] stated that while there is a huge set of algorithms proposed under the genetics-based ruleset algorithm umbrella, there was no explicit framework where all of these algorithms could be categorized, and no exhaustive comparisons of the performance of these algorithms. The authors' motivation was to provide a state of the art of genetics-based machine learning (GBML) algorithms for rule induction, proposing a taxonomy that defined a general framework where the algorithms could be placed. The author's of that work treated decision tree algorithms as rule induction algorithms. As already seen in previous sections of this chapters, rule-sets and decision trees are close both in how they are built, sometimes trees being an early step in the rule induction algorithm, they are similarly represented, and both algorithm types have explaining capacity. The experiments in this thesis follow the same experimental methodology, using the same datasets, the same train and test partitions and, comparing the results of the contributions to the results published by that article.

Classically, evolutionary rule-based algorithms were classified the Michigan-style and the Pittsburgh-style groups, but many algorithms could not be easily classified in any of the two. Thus, this article proposed a taxonomy based on the representation of the chromosome of the associated evolutionary algorithm. The taxonomy preserves the classical Michigan and Pittsburgh categories and defines new categories to encompass all evolutionary rule-based approaches. Michigan approaches encode each rule as individuals, whereas Pittsburgh style algorithms encode whole rulesets as individuals [134].

The article proposed a hierarchical comparison to focus on algorithms with better performance, by first selecting the best performing algorithm on each category to act as representative, and the performing a cross-category comparison of the representatives and some well-known non-evolutionary algorithms for rule induction.

All GBML algorithms for rule induction codify the rule or ruleset in a chromosome and use an evolutionary algorithm as the search mechanism. However there are many differences among these algorithms such as the type of learning scheme, the ruleset representation, the representation of each rule, and the design of the evolutionary algorithm that guides the search.

The taxonomy proposed by this article divides algorithms in three categories based on the chromosome codification, further dividing one of the categories into three subcategories for a total of five families.

1. The algorithms in the first category encode each rule in a single chromosome and the population evolves to find the best ruleset. This category is formed by three subcategories.

    (a) The first subcategory includes the algorithms following the Michigan approach, where a ruleset is increasingly updated through the sequential observation of training examples and their classification. The ruleset is improved by the action of the evolutionary algorithm.

The algorithms used in the empirical study that belong to this family are XCS [152] and UCS [20].

(b) The second subcategory includes algorithms following the iterative rule learning (IRL) approach, where the evolutionary algorithm learns rule by rule iteratively, and removes the examples covered by the ruleset from the training sample by following the separate-and-conquer rule induction paradigm. The algorithms used in the empirical study that belong to this family are SIA [147] and HIDER [3].

(c) The third subcategory includes algorithms following the genetic cooperative learning (GCCL) approach, where all rules/chromosomes are evolved together in the genetic algorithm. The chromosomes cooperate with each other to perform the classification task but the final ruleset does not need to include all rules. The rules compete with each other to be present at the final ruleset. The algorithms used in the empirical study that belong to this family are CORE [143], OCEC [93] and COGIN [71].

2. The algorithms in the second category encode a ruleset in a chromosome. These algorithms maintain a set of rulesets and in the end, the ruleset with the best discriminating capacity is kept. This category is also known as the Pittsburgh approach. The algorithms used in the empirical study that belong to this family are GIL [89], Pitts-GIRLA [37], DMEL [14], GAssist [15], OIGA [161], and ILGA [72].

3. The algorithms in the third category encode a decision tree or tree rule on a chromosome. The idea is to use the genetic algorithm to search a highly accurate tree. Decision trees can be interpreted as ruleset by looking the decisions on each branch of the tree as the decisions that make up a rule. The author's defined these approaches ad hybrid evolutionary decision trees (HEDT). The algorithms used in the empirical study that belong to this family are DT-GA [27], Oblique-DT [26], and TARGET [70].

Table 2.7 summarizes the GBML algorithms used in the empirical study.

The classical non-evolutionary algorithms compared to the genetics-based algorithm were CART [22], AQ [107], CN2 [34], C4.5 [128], C4.5rules [128] and RIPPER [36].

1. Classification And Regression Tree or CART: This algorithm creates binary trees by recursively partitioning nodes into two subnodes. It uses the Gini-index as the split criterion.

2. Automatic Qualifier or AQ: This algorithm follows the separate-and-conquer strategy that implements the START method of inductive learning.

3. CN2: induces an ordered list of rules using entropy as the search heuristic. It also works in a separate-and-conquer fashion by iteratively creating rules that cover large amounts of examples from a single class and as few as possible from others.

4. C4.5: a decision tree algorithm that creates trees from the top down by selecting the attribute with the best gain ratio: a normalized information gain which in turn is the difference in entropy between the node and the possible split. This algorithm has already been discussed in depth in Section 2.4.

5. C4.5rules: Rule induction algorithm based on C4.5. Rules are extracted from the C4.5 tree and a Hill-Climbing search algorithm is used to find the best subset of rules according to their minimum description length (MDL).

6. RIPPER: This rule induction algorithm also uses the separate-and-conquer strategy by sequentially creating rulesets targeting one class at a time, and pruning each rule right after creating it.

Table 2.8 summarizes the parameters for GBML algorithms whereas Table 2.9 does the same for the classical non-evolutionary algorithms.

The empirical study used 96 datasets divided into three contexts. The first context consists of 30 standard datasets with a wide range of characteristics (dataset size, number of classes, class distribution) that represent all types of classification problems. The second context is concerned with the class imbalance problem, a challenge in classification already discussed on Section 2.3. Most of these datasets were obtained processing multi-class datasets by grouping one or more classes into the minority class and the rest into the majority class. The third context consists of the same datasets as the second context but oversampling the minority class using SMOTE until the training sample was balanced. These datasets are described at length at the beginning of Part III.

In this article, different performance metrics were used to compare the algorithms over different classification contexts.

For the standard dataset classification context two performance metrics were used. The classification rate or overall accuracy, and Cohen's kappa. These metrics were used because their simplicity, low computational cost and successful application to two-class and multi-class problems.

For the second and third classification contexts a metric suited for class-imbalance was used as the accuracy does not take into account the class distribution of the dataset. This metric was geometric mean (GM) of true-positive (TP) and true-negative (TP) rates.

The results of this study are found in Appendix B. In order to support the results of the experiments, the article used statistical tests for significance. Specifically, it used the Wilcoxon signed rank test [151] for pairwise comparisons and the Friedman test with the Iman-Davenport extension [58, 59] for tests comparing multiple algorithms. If the Friedman test found significant differences, the Shaffer [138] post hoc test was used to find in which of the $n \times n$ pairwise comparisons differences were found.

Figure 2.10 shows the best ranking algorithms for each GBML subcategory in the taxonomy for every classification context. When comparing against classical non-evolutionary algorithms the best ranking algorithms were XCS,

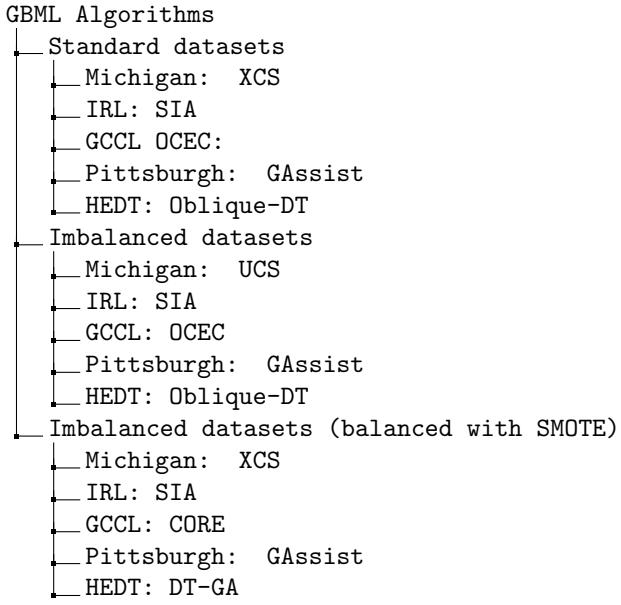

Figure 2.10: Best ranking algorithms for each subcategory by classification context.

GAssist and C4.5 for standard datasets; RIPPER, C4.5rules and Oblique-DT for imbalanced datasets; and XCS, GAssist and C4.5 for imbalanced datasets preprocessed with SMOTE. An extra experiment comparing the performance of each algorithm using and not using SMOTE to preprocess the training data showed that almost every algorithm (all except CART and RIPPER) achieved significant improvement in their discriminating capacity in terms of GM.

The article concluded that XCS and GAssist were the two best performing algorithms among the compared GBML algorithms. Their good performance is due to interactions between several components and the good performance cannot be easily attributed to a single component of the learning process. Both algorithms achieved competitive results for standard classification in terms of accuracy. XCS's models are not very interpretable as XCS ran populations of 6400 rules. GAssist presented a good trade-off between accuracy and interpretability. This interpretability was not, measured, reported in the article. XCS's performance dropped on raw imbalanced datasets (in fact, UCS ranked best for the Michigan family on imbalanced datasets), however, the authors note that this might be due to that particular implementation of XCS. For non-preprocessed imbalanced datasets non-evolutionary seem to get the upper hand in contrast to the other classification contexts where the best positions belonged to GBML algorithms. The authors hypothesized that this could be because these algorithms use the overall accuracy of the classifier as the fitness function and aim to maximize this measure at the cost of the correct classification of the minority

classes.

To the knowledge of the article's authors this was the first exhaustive experimental study comparing state-of-the-art and former GBML algorithms with non-evolutionary algorithms. At the time of writing it has been cited 152 times according to Google Scholar, and is considered a reference work in this field due to its scope and relevance. The authors of the article encouraged other researchers to use their empirical to compare the discriminating capacity of future proposals, and, as mentioned at the beginning of this section this article serves as methodological base for this thesis and is heavily referenced throughout the thesis as the "reference work".

Algorithm	Acronym	Family	Reference
XCS	XCS	Michigan	[152]
UCS	UCS	Michigan	[20]
Supervised Inductive Algorithm	SIA	IRL	[147]
Hierarchical Decision Rules	HIDER	IRL	[3]
Co-evolutionary Rule Extractor	CORE	GCCL	[143]
Organizational co-evolutionary algorithm for classification	OCEC	GCCL	[93]
Coverage-based genetic induction	COGIN	GCCL	[71]
Genetic-based Inductive Learning	GIL	Pittsburgh	[89]
Pittsburgh Genetic Interval Rule Learning Algorithm	Pitts-GIRLA	Pittsburgh	[37]
Data Mining for Evolutionary Learning	DMEL	Pittsburgh	[14]
Genetic Algorithms based Classifier System Ordered Incremental Genetic Algorithm	GAssist	Pittsburgh	[15]
Ordered Incremental Genetic Algorithm	OIGA	Pittsburgh	[161]
Incremental Learning with Genetic Algorithms	ILGA	Pittsburgh	[72]
Hybrid Decision Tree - Genetic Algorithm	DT-GA	HEDT	[27]
Oblique Decision Tree	Oblique-DT	HEDT	[26]
Tree Analysis with Randomly Generated and Evolved Trees	TARGET	HEDT	[70]

Table 2.7: Genetics-based machine leaning algorithms for rule induction.

Michigan Algorithms	
Algorithm	Parameters
XCS	Number of explores = 100 000, population size = 6400, $\alpha = 0.1$, $\beta = 0.2$, $\delta = 0.1$, $\nu = 10.0$, $\theta_{mna} = 2$, $\theta_{del} = 50.0$, $\theta_{sub} = 50.0$, $\in_0 = 1$, do Action Set Subsumption = false, fitness reduction = 0.1, $p_I = 10.0$, $F_I = 0.01$, $\in_I = 0.0$, $\gamma = 0.25$, $\chi = 0.8$, $\mu = 0.04$, $\theta_{GA} = 50.0$, doGASubsumption = true, type of selection = Roulette wheel selection (RWS), type of mutation = free, type of crossover = 2 point, $P_{\#} = 0.33$, r0 = 1.0, m0 = 0.1, $l_0 = 0.1$, doSpecify = false, nSpecify = 20.0, pSpecify = 0.5
UCS	Number of explores = 100 000, population size = 6400, $\delta = 0.1$, $acc_0 = 0.99$, $P_{cross} = 0.8$ $P_{mut} = 0.04$, $\theta_{GA} = 50.0$, $\theta_{del} = 50.0$, $\theta_{sub} = 50.0$, doGASubsumption = true, $r_0 = 0.6$, type of selection = RWS, type of mutation = free, type of crossover = 2 point, $m_0 = 0.1$
IRL Algorithms	
Algorithm	Parameters
SIA	Number of iterations = 200, $\alpha = 150$, $\beta = 0$, threshold strength = 0
HIDER	Pop. size = 100, number of gen. = 100, mutation prob.= 0.5 percentage of crossing = 80, extreme mutation prob. = 0.05, Prune examples factor = 0.05, penalty factor = 1, error coefficient = 0
GCCL Algorithms	
Algorithm	Parameters
CORE	Pop. size = 100, co-population size = 50, gen. limit = 100, number of co-populations = 15, crossover rate = 1.0, mutation prob. = 0.1, regeneration prob. = 0.5
OCEC	Number of total generations = 500, number of migrating/exchanging members = 1
COGIN	Misclassification error level = 2, gen. limit = 1000, crossover rate = 0.9, negation bit = yes
Pittsburgh Algorithms	
Algorithm	Parameters
GIL	Pop. size = 40, number of gen. = 1000, $w_1 = 0.5$, $w_2 = 0.5$, $w_3 = 0.01$, rules exchange = 0.2, rule exchange selection = 0.2, rules copy = 0.1, new event = 0.4, rules generalization = 0.5, rules drop = 0.5, rules specialization = 0.5, rule split = 0.005, nominal rule split = 0.1, linear rule split = 0.7, condition drop = 0.1 conjunction to disjunction = 0.02, introduce condition = 0.1, rule directed split = 0.03, reference change = 0.02, reference extension = 0.03, reference restriction = 0.03, condition level prob. = 0.5, lower threshold = 0.2, upper threshold = 0.8
Pitts-GIRLA	Number of rules: 30, number of generations: 10 000, population size: 61 chromosomes, crossover probability: 0.7, mutation probability: 0.5
DMEL	Pop. size = 30, crossover prob. = 0.5, mutation prob. = 0.0001, number of gen. = 1000
GAssist	Threshold in hierarchical selection = 0, iteration of activation for rule deletion operator = 5, iteration of activation for hierarchical selection = 24, minimum number of rules before disabling the deletion operator = 12, minimum number of rules before disabling the size penalty operator = 4, number of iterations = 750, initial number of rules = 20, population size = 400, crossover probability = 0.6, probability of individual mutation = 0.6, probability of value 1 in initialization = 0.90, tournament size = 3, possible size in *micro-intervals* of an attribute = 4, 5, 6, 7, 8, 10, 15, 20, 25, maximum number of intervals per attribute = 5, $p_{split} = 0.05$, $p_{merge} = 0.05$, probability of reinitialize begin = 0.03, probability of reinitialize end = 0, Use MDL = true, iteration MDL = 25, initial theory length ratio = 0.075, weight relaxation factor = 0.90, class initialization method = cwinit, default class = auto
OIGA	Crossover prob. = 1.0, mutation prob. = 0.01, pop. size = 200, number of rules = 30, stagnation = 30, generations = 200, survivors percent = 0.5, attribute order = descendent
ILGA	Crossover prob. = 1.0, mutation prob. = 0.01, pop. size = 200, number of rules = 30, stagnation = 30, generations = 200, survivors percent = 0.5, attribute order = descendant, crossover reduction rate = 0.5, mutation reduction rate = 0.5, incremental strategy = 1
Hybrid Evolutionary Decision Trees	
Algorithm	Parameters
DT-GA	Confidence = 0.25, instances per leaf = 2, threshold S to consider a small disjunct = 10, number of gen. = 50, crossover prob. = 0.8, mutation prob. = 0.01, examples threshold = 5
Oblique-DT	Number of total generations for the GA = 25, population size = 20 $\sqrt{d}$ (d = dimensionality)
TARGET	Split prob. = 0.5, number of gen. = 100, number of trees = 50, number of crossovers = 30, number of mutations = 10, number of clones = 5, number of transplants = 5

Table 2.8: Parameter specification for the genetics-based algorithms.

Algorithm	Parameters
CART	Max depth of the tree = 90
AQ	Star size = 5
CN2	Star size = 5
C4.5rules	Prune = true, confidence level = 0.25, minimum number of item-sets per leaf = 2
C4.5	Prune = true, confidence level = 0.25, minimum number of item-sets per leaf = 2
RIPPER	Size of growing subset = 66%, Repetitions of the optimization stage = 2

Table 2.9: Parameter specification for the classical algorithms.

Part III

Contributions

In this Part, the contributions of the thesis are presented. A section dedicated to the description of the used datasets is followed by the two chapters grouping the distinct contributions of the thesis: contributions to the consolidation of decision tree algorithms and contributions to PART-like ruleset induction algorithms. A final section will compare the performance of both contribution groups.

KEEL Datasets

With the exception of a study found in Appendix H, the entire thesis used the same datasets. These are the same datasets used by the reference work were taken from the KEEL repository [8]. The 96 datasets used in this thesis were divided into three classification contexts.

The first classification context is standard classification. It is composed of 30 standard datasets that present a broad range of characteristics: two-class and multi-class datasets (up to 22 classes for *abalone*), small and big datasets, balanced and very imbalanced datasets, different attributes, etc. The characteristics of these datasets are described on Table Datasets.1, ordered by the proportion of the minority class in ascending order.

The second context is the context of imbalanced classification and is composed of 33 datasets that represent a specific problem within classification, the problem of class imbalance in two-class datasets. Some of these datasets are two-class versions of the datasets from the first context. The characteristics of these datasets are described on Table Datasets.2. The datasets in these tables are ordered by the relative size of the minority classes.

Finally, the third classification context consists of the same datasets of the second context, but oversampling the minority class in the training sample using SMOTE [31] until the two classes are balanced.

		#features								
		#discrete								
dataset	#examples	#nominals	#ordinals	#continuous	#total	#classes	%min. class	%maj. class	size of min. class	size of maj. class
nursery	1296	8	0	0	8	5	0.08%	33.34%	1	432
abalone	418	1	0	7	8	22	0.24%	16.51%	1	69
ecoli	336	0	0	7	7	8	0.6%	42.56%	2	143
lymphography	148	15	0	3	18	4	1.36%	54.73%	2	81
car	1728	0	6	0	6	4	3.77%	70.03%	65	1210
zoo	101	16	0	0	16	7	3.97%	40.6%	4	41
flare	1066	9	2	0	11	6	4.04%	31.06%	43	331
glass	214	0	0	9	9	6	4.21%	35.52%	9	76
cleveland	297	0	0	13	13	5	4.38%	53.88%	13	160
dermatology	358	3	30	0	33	6	5.59%	31.01%	20	111
balance	625	0	0	4	4	3	7.84%	46.08%	49	288
penbased	1100	0	0	16	16	10	9.55%	10.46%	105	115
newthyroid	215	0	0	5	5	3	13.96%	69.77%	30	150
hepatitis	80	13	0	6	19	2	16.25%	83.75%	13	67
contraceptive	1473	3	0	6	9	3	22.61%	42.71%	333	629
vehicle	846	0	0	18	18	4	23.53%	25.77%	199	218
haberman	306	0	0	3	3	2	26.48%	73.53%	81	225
wine	178	0	0	13	13	3	26.97%	39.89%	48	71
breast	277	5	4	0	9	2	29.25%	70.76%	81	196
german	1000	11	4	5	20	2	30%	70%	300	700
iris	150	0	0	4	4	3	33.34%	33.34%	50	50
wisconsin	630	9	0	0	9	2	34.61%	65.4%	218	412
tictactoe	958	9	0	0	9	2	34.66%	65.35%	332	626
pima	768	0	0	8	8	2	34.9%	65.11%	268	500
magic	1902	0	0	10	10	2	35.13%	64.88%	668	1234
bupa	345	0	0	6	6	2	42.03%	57.98%	145	200
heart	270	5	2	6	13	2	44.45%	55.56%	120	150
australian	690	6	0	8	14	2	44.5%	55.51%	307	383
crx	653	9	0	6	15	2	45.33%	54.68%	296	357
ring	740	0	0	20	20	2	49.6%	50.41%	367	373
Mean	638.93	4.07	1.6	6.1	11.77	4.27	21%	50%	139	319.93
Median	521.5	0.5	0	6	9.5	3	23%	54%	73	209

Table Datasets.1: KEEL datasets for standard classification (first context).

	#features				second context				third context	
	#discrete									
dataset	#nominals	#ordinals	#continuous	#total	#examples	% min. class	size of min. class	size of maj. class	#examples	size per class
abalone19	1	0	7	8	4174	0.77%	32	4142	8284	4142
yeast6	0	0	8	8	1484	2.49%	37	1447	2894	1447
yeast5	0	0	8	8	1484	2.96%	44	1440	2880	1440
yeast4	0	0	8	8	1484	3.43%	51	1433	2866	1433
yeast2_vs_8	0	0	8	8	482	4.15%	20	462	924	462
glass5	0	0	9	9	214	4.2%	9	205	410	205
abalone9_vs_18	1	0	7	8	731	5.65%	41	690	1380	690
glass4	0	0	9	9	214	6.07%	13	201	402	201
ecoli4	0	0	7	7	336	6.74%	23	313	626	313
glass2	0	0	9	9	214	8.78%	19	195	390	195
vowel0	2	0	11	13	988	9.01%	89	899	1798	899
page-blocks0	0	0	10	10	5472	10.23%	560	4912	9824	4912
ecoli3	0	0	7	7	336	10.88%	37	299	598	299
yeast3	0	0	8	8	1484	10.98%	163	1321	2642	1321
glass6	0	0	9	9	214	13.55%	29	185	370	185
segment0	0	0	19	19	2308	14.26%	329	1979	3958	1979
ecoli2	0	0	7	7	336	15.48%	52	284	568	284
new-thyroid1	0	0	5	5	215	16.28%	35	180	360	180
new-thyroid2	0	0	5	5	215	16.89%	36	179	358	179
ecoli1	0	0	7	7	336	22.92%	77	259	518	259
vehicle0	0	0	18	18	846	23.64%	200	646	1292	646
glass0-1-2-3_vs_4-5-6	0	0	9	9	214	23.83%	51	163	326	163
haberman	0	0	3	3	306	27.42%	84	222	444	222
vehicle1	0	0	18	18	846	28.37%	240	606	1212	606
vehicle2	0	0	18	18	846	28.37%	240	606	1212	606
vehicle3	0	0	18	18	846	28.37%	240	606	1212	606
yeast1	0	0	8	8	1484	28.91%	429	1055	2110	1055
glass0	0	0	9	9	214	32.71%	70	144	288	144
iris0	0	0	4	4	150	33.33%	50	100	200	100
pima	0	0	8	8	768	34.84%	268	500	1000	500
ecoli0_vs_1	0	0	7	7	220	35%	77	143	286	143
wisconsin	9	0	0	9	683	35%	239	444	888	444
glass1	0	0	9	9	214	35.51%	76	138	276	138
Mean	0.39	0	9	9.39	919.94	17.61%	120	799.94	1599.88	799.94
Median	0	0	8	8	482	15.48%	52	444	888	444

Table Datasets.2: KEEL datasets for imbalanced classification (second and third contexts).

Chapter 3

Contributions to the consolidation of decision tree algorithms

3.1 Introduction

This chapter focuses on the contributions made during this thesis to the consolidation methodology applied to decision tree algorithms.

On the one hand, as described in Section 2.6, the consolidation methodology has been successfully applied to the C4.5 decision tree algorithm. This methodology could be applied to other classification algorithms, and especially, to other decision tree induction algorithms. One such candidate is the C4.4 algorithm [125], a Probability Estimation Tree (PET) variant of the C4.5 algorithm where no collapsing or pruning is performed, and the a posteriori probabilities of the leaves are corrected. Another possible candidate for consolidation is CHAID, a widely-known decision tree algorithm. However, CHAID has a notable limitation compared to C4.5: the inability to handle continuous variables. Thus, this chapter introduces a new variant of the CHAID algorithm, named CHAID*, that can handle continuous variables. Also, mirroring the C4.4 algorithm, the CHAIC algorithm is also proposed, by modifying CHAID* the same way C4.5 was modified to create the C4.4 algorithm. These three algorithms have also been consolidated.

On the other hand, until recently, the CTC algorithm had two parameters (apart from the ones used by the base algorithm), the number of subsamples and the type of resampling used. Previous research mostly used stratified subsamples, keeping the original class distribution of the sample, and a set of fixed values for the number of samples, ranging between three and two hundred. Recent research suggests that changing the class distribution of the samples is beneficial, with balanced class distributions garnering the best results. How-

ever, unlike with stratified samples that represent the same percentage of each class, changing the class distribution means that a subsample represents each class to a different degree. In consequence, this means that when using balanced samples, whereas a specific number of samples might be enough for a dataset with a not too imbalanced distribution, it might fall short of representing all classes properly in cases where the imbalance is greater. This chapter proposes a new strategy that replaces the fixed number of subsamples strategy.

Finally, it is known that in presence of the class imbalance problem, where the consolidation methodology was born, the performance of some classification algorithms is detrimented. In the case of C4.5, and by extension CTC, the pruning process can be greatly affected. The pruning process of decision trees is meant to avoid overfitting by removing the most specialized branches of the tree. However, in the presence of class imbalance, this could mean branches that identify the minority class. It is not uncommon for decision trees to end up fully pruned into just the root node. These root-trees, even though they have a relatively high accuracy, simply classify all examples as the majority class, and thus, offer no explanation. This chapter proposes and analyzes the use of the Not Root Tree (NRT) strategy, using pruning only when it does not fully prune the tree, and compares the performance of base and consoldidated decision trees with different pruning strategies in the presence of class imbalance.

The structure of this chapter is as follows, Section 3.2, Section 3.3 and Section 3.4 describe the contributions presented in this chapter. Section 3.5 outlines the experimental setup for the experiments in this chapter, Section 3.6 analyzes the results of the experiment and, finally, Section 3.7 makes a brief summary.

3.2 New base algorithms

This section covers the two new decision tree algorithms proposed in this thesis that also serve as base algorithms for consolidation.

3.2.1 CHAID*

The CHAID (Chi-squared Automatic Interaction Detector) is a decision tree algorithm described in Section 2.4. This decision tree algorithm can only handle discrete variables, and, since all but one datasets used in this thesis contain continuous or numeric values, this algorithm could not have been used. CHAID* is a variation of CHAID developed to cope with that constraint.

When a split is to be made, CHAID* calculates the best split for each variable. Discrete variables are treated in the same way they are treated in CHAID. The best split is found by merging variable values into groups using the chi-squared as the split criterion. Continuous variables are handled with a hybrid between how CHAID handles discrete variables and how C4.5 handles continuous variables. The split criterion is also used for splits on continuous variables, and these variables are split into two branches. The values the variable

can take are taken into consideration as potential split points and each potential split is tested using the chi-squared test to look for the most significant split, if any. The p-value of the potential continuous splits is also adjusted using the same Bonferroni Coefficient in order to make fair decisions.

In C4.5 missing values are assigned to all branches of a node, readjusting their weight to the weight of each branch. In CHAID*, the split is first decided without taking the missing values into account. If the split is on a discrete variable, examples with missing values are placed in a single branch following the procedure explained in Section 2.4.2. If the split is on a continuous variables, the algorithm will attempt to put the missing values into one of the two branches using the chi-squared test in the same way it is used with discrete variables. If this is not possible they will be placed in a third branch of their own.

CHAID has mechanisms to stop the tree construction phase even if the leaf nodes are not fully homogeneous. These are referred to *pre-pruning* strategies, but it does not have mechanisms to reduce the final tree size. Initial experiments showed that decision trees built by CHAID were significantly more complex than those built by other decision tree algorithms, and specifically C4.5. Too complex decision trees tend to overfit to the training data. Also in this thesis, one of the aspects of how algorithms are compared against each other is the structural complexity of the models they create, so it was decided to add a pruning process to CHAID*. CHAID* uses the same pruning mechanisms used by C4.5 and explained in Section 2.4.1.2. Since this pruning process is related to the node's and subtree's error rate and not to the split criterion of the algorithm. It does not need a special adaptation to this algorithm.

3.2.2 CHAIC

CHAIC is to CHAID* what C4.4 is to C4.5. The aim of proposing this algorithm is to test if same changes from C4.5 to C4.4 could also help improve the results of CHAID* decision trees. The name of this new proposal is a play on words in line with C4.4.

The tree construction phase is the same on as in CHAID*. The split criterion is the chi-squared test, variables are treated in the same way as in CHAID* and the same *pre-pruning* criteria are used.

The differences appear on how the tree is treated once it is built. CHAID* prunes the trees using the same pruning procedure as C4.5. CHAIC treats trees the same way C4.4 does. There is no pruning of the tree, so the size and, in consequence, the structural complexity stay the same as unpruned CHAID* trees. However, the class distribution probabilities on leaf nodes are adjusted using the Laplace correction.

3.3 *Coverage*-based resampling

Creating balanced subsamples without oversampling the minority class requires undersampling the majority class. As a consequence, resampling imbal-

anced datasets results in subsamples where the majority class is only represented in a small proportion. Previously used number of samples (N_S) values, ranging from 3 to 200, might not be enough to cover a significant portion of the majority class in imbalanced datasets, while the highest values are completely unnecessary and time consuming for the most balanced problems. *Coverage*-based resampling is a strategy that adapts to the class distribution present in the dataset and computes a particular N_S for a dataset and coverage value.

Coverage represents the probability of any example in the training sample being in at least one subsample; in other words, probability of the entire training sample being represented by the randomly undersampled subsamples. When subsamples have a different class distribution from the original training sample each class is represented to a different degree in a subsample. This degree depends on the use of replacement when creating subsamples and the ratio between the number of examples from a class in the subsamples and the number of examples from the same class in the original sample. Depending on this ratio, the subsampling process favors some classes more than others. The *least favored class* is defined as the one with the lowest ratio of instances in the subsamples to instances in the training sample. Since the goal of coverage is to ensure a minimum representation for all classes, the probability of an example from the least favored class being present in a given subsample will be used as a basis. The probability for examples from any other class will be higher than that.

For example, let us consider the datasets used in this thesis, specifically the context regarding imbalanced two-class datasets. For the experiments in this chapter the subsamples are fully balanced by randomly undersampling the majority class without replacement. One of the used subsample sizes is the double of the number of minority class examples in the original training sample. This means that each of these subsamples fully covers the minority class. However, each subsample is not able to cover the majority class entirely, and the greater the class imbalance present in the original training sample the greater number of subsamples that will be needed to cover a specific percentage of examples. Figure 3.1 shows how different subsample numbers translate into different coverage values. In this figure, balanced subsamples are created by randomly undersampling the majority class from a training sample with 33% of minority class (2:1 Imbalance Ratio, meaning there are two instances of the majority class per instance of the minority class) examples.

Let t_{lfc} be the probability of an example of the least favored class (*lfc*) being present in a subsample. As mentioned, this probability is influenced by the type of subsampling. The subsamples used in this chapter modify the class distribution, do not use replacement, and the probability is calculated by dividing the number of examples of the least favored class in a subsample by the number of examples of the least favored class in the training sample.

$$t_{lfc} = \frac{lfClassExamplesOnSubsample}{lfClassExamplesOnOriginalSample}$$

Then, the probability of an example of the *lfc* not being present in a sub-

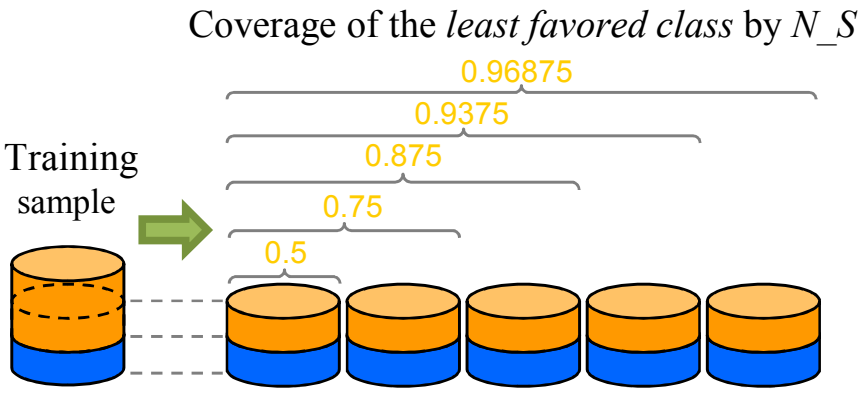

Figure 3.1: Example of the coverage achieved with multiple balanced subsamples generated by randomly undersampling a dataset with a 2:1 Imbalance Ratio.

sample is:

$$1 - t_{lfc}$$

The generation of each subsample is an independent event, so the probability of an example not being in any of the *N_S* subsamples is:

$$(1 - t_{lfc})^{N\_S}$$

Thus, the probability of an example being in at least one of *N_S* subsamples or the coverage is:

$$coverage = 1 - (1 - t_{lfc})^{N\_S}$$

Since the goal is to work with fixed coverage values and the number of subsamples *N_S* for a specific coverage has to be computed, the following has to be solved:

$$N\_S = \lceil \log_{1-t_{lfc}}(1 - coverage) \rceil$$

As an example, the *abalone19* imbalanced two-class dataset has 4174 examples. The experiments use a 5-fold cross validation, meaning that 4/5 of the sample go into the training sample and the other 1/5 of the sample is used as the test sample each time. For this dataset the training sample has 3340 examples. The class distribution for the minority class is 0.77% in this case, meaning that 26 examples in the training sample belong to the minority class while the other 3314 belong to the majority class. A subsample type used in this chapter has the same number of examples for each class, which is the size of the minority class in the training sample. Thus, in this case the subsamples have 52 examples (26 from each class), so a subsample has all the minority class examples but only a few of the majority class examples. That makes the majority class the least favored class. Thus, in this case *lfClassExamplesOnSubsample* is 26

while *lfClassExamplesOnOriginalSample* is 3314. From this t_{lfc} is calculated to be 0.0078. Thus, for a coverage of 99% (i.e. 0.99), for the *abalone19* dataset 585 subsamples would be required.

The use of coverage is not only limited to balanced samples. *Coverage*-based resampling can be used with any type of sample to determine the number of samples needed.

One such type would be bootstrap samples used by Bagging. Bootstrap samples are the same size as the original sample but with replacement. Meaning that each type an instance from the original sample is selected, it is 'replaced', and can be drawn again. When creating this type of sample the dataset's class distribution is not taken into account. Bauer and Kohavi [17] already published the equivalent for t_{lfc} for bootstrap samples.

Each example of the original sample has a probability of $1/n$ of being selected each time an instance is picked, n being the size of the sample. Thus, the probability of an example not being picked each time an instance is selected is $1-1/n$. For a single bootstrap sample, the probability an instance has of being selected for a bootstrap sample is $1-(1-1/n)^n$. For large n values this is approximately $1-1/e$ or 63.2%. As the goal of coverage is to determine the probability of an example being present in a group of N_S samples, we get the following equation:

$$coverage = 1 - 1/e^{N\_S} = 1 - e^{-N\_S}$$

In order to solve for N_S, the equation has to be transformed into:

$$N\_S = \lceil -\ln(1 - coverage) \rceil$$

Table 3.1 shows some examples of N_S values for all coverage values calculated for some of the most imbalanced and some of the most balanced datasets from Table Datasets.2.

Even though the notion of coverage and coverage-based resampling arise from the need to address the changes in class distribution, stratified samples CTC used until now could also make use of coverage-based resampling. These samples keep the sample's class distribution and do not use replacement, which means that no class is favored over others when undersampling. This results

	% min.	Coverage									
Dataset	class	10%	20%	30%	40%	50%	75%	90%	95%	99%	99.9%
abalone19	0.77%	14	29	46	65	89	177	293	381	585	878
yeast6	2.49%	5	9	14	20	27	53	88	115	176	264
yeast5	2.96%	4	8	12	17	22	44	73	95	146	218
yeast1	28.91%	3	3	3	3	3	3	5	6	9	14
iris0	33.33%	3	3	3	3	3	3	4	5	7	11
glass1	35.51%	3	3	3	3	3	3	3	4	6	9

Table 3.1: Examples of coverage-based N_S values for some datasets using *maxSize* subsamples.

in all classes being equally represented in a subsample and the probability of an example being selected for a subsample is equal for all classes and only dependent on the r size ratio between a subsample and the original sample. In order to calculate a N_S value for a stratified sample of certain size the following has to be solved:

$$N\_S = \lceil \log_{1-r}(1 - coverage) \rceil$$

In previous research [12] it has been observed that CTC works best when most of the information of the original sample is covered by the set of subsamples. Therefore, this chapter hypothesizes that CTC will work best with high coverage values that minimize the loss of information when resampling.

3.4 New consolidated algorithms

This section describes how consolidation has been applied to the C4.4 algorithm and the newly proposed algorithms: CHAID* and CHAIC. As all of these algorithms are decision tree algorithms, the consolidation is applied on each of the decision tree nodes. The process has already been described in Section 2.6.1.As all four algorithms are similar, the consolidation of these algorithms is also similar.

Originally, the *CTC* name referred to the consolidation of the C4.5 algorithm. But, as all four consolidated algorithm could share the CTC (Consolidated Tree Construction) name, from now CTC45 will be used to refer to the original CTC algorithm, and CTC44, CTCHAID (without the * from CHAID*'s name) and CTCHAIC to refer to the new consolidated algorithms.

3.4.1 Consolidation of continuous variables in CTC45, CTC44, CTCHAID and CTCHAIC

If the most voted variable is continuous, the median value of all proposed cut-point values is used as the consolidated cut-point. The examples with a value lower or equal to the cut-point are placed on one branch, and examples with a value greater than the cut-point are placed in another. In CTC45 and CTC44, the examples with missing values on continuous variables are treated in the same way C4.5 does, whereas in CTCHAID and CTCHAID they are treated as CHAID* does.

3.4.2 Consolidation of discrete variables in CTC45 and CTC44

If the most voted variable in CTC45 or CTC44 is discrete, the behavior is the same as in C4.5 and C4.4: a branch is created for every possible value the variable can take. The examples with missing values on discrete variables are treated in the same way C4.5 does.

3.4.3 Consolidation of discrete variables in CTCHAID and CTCHAIC

Before the work presented in this thesis, consolidation was only applied to C4.5. By default, C4.5 creates a branch for any value a discrete variable can take. However CHAID and CHAID* group different values on the same branch by using the algorithm developed by Kass and mentioned in Section 3.2.1. This thesis proposes a way to consolidate the stratification of discrete variables. If the most voted variable in CTCHAID or CTCHAIC is discrete, each of the samples passes on the contingency table for that variable. An average table is created, and this table is passed on to the algorithm developed by Kass to find the most significant value groupings for that variable. The examples with missing values on discrete variables are treated in the same way CHAID and CHAID* do.

3.5 Experimental Setup

The three experiments in this chapter follow a very similar structure to the reference work as the results of the contributions are compared to the results of the reference work.

The reference work used the same 96 datasets, described at the beginning of Part III, divided into the same three classification contexts and the same train and test partitions for the cross validation, and the same performance metrics are used. The results published by the reference work can be found on Appendix B.

This chapter contains the results of three different experiments. The first experiment compares the performance of the original CTC45 algorithm using two different balanced subsample sizes and a number of coverage values.

The used coverage values are 10%, 20%, 30%, 40%, 50%, 75%, 90%, 95%, 99% and 99.9%. One last value, which technically is not a coverage value, is the static value of 3 subsamples (N_S=3), which is the smallest possible number consolidated algorithms can use. Thus, whenever the number of subsamples for a dataset, sample size and coverage value falls below three, three samples are used.

The first of the subsample sizes used is referred to as *sizeOfMinClass* where subsamples have the size of the minority class in the original sample. This type of sample was proposed by Weiss [150]. In this type of sample each class has n/k examples where n is the size of the sample and k is the number of classes. In the second type of subsample, *maxSize*, each class has the size of the minority class in the original training sample. This is the biggest possible balanced subsample without oversampling the minority class and is also the most widely used strategy when random undersampling is applied.

For these experiments, where subsamples have a balanced class distribution, the least favored class used to compute the number of subsamples for a coverage is the majority class. The subsample size is that of the minority class in the training sample for one of the subsamble sizes, and the size of the minority class

multiplied by the number of classes for the other size of subsample. There are two exceptions to this rule:

- The standard datasets are mostly multi-class. Sometimes the least populous class (the minority class) has as little as one example. This makes *sizeOfMinClass* subsamples impossible and *maxSize* subsamples too small. Because of this, a rule has been put in place to ensure that the number of minority class examples found in a subsample is at least equal the 1% of the total number of examples in the dataset when creating *sizeOfMinClass* subsamples and 2% when creating *maxSize* subsamples. In these cases random oversampling is applied to the minority class until this minimum is reached. Only three datasets out of thirty need this exception to the general rule: *abalone*, *car* and *ecoli*.

- The *iris* standard dataset is unique because it is fully balanced. It is composed of three classes, each with 50 examples. Using the general rule to build *maxSize* subsamples, a single subsample would already contain all of the examples of the original training sample, resulting in a coverage of 100%. In this case, in order to obtain a set of different subsamples to build the consolidated trees, the size of the subsamples is reduced in order to guarantee that the subsamples will be stratified: i.e. the subsamples will also be balanced. Thus, in the case of the *iris* standard dataset the *maxSize* subsample size will be 55% of the training sample's. This specific ratio is chosen because it is the average ratio between subsamples and the original training sample for all datasets.

Table 3.2 and Table 3.3 summarize the subsample numbers for standard and imbalanced datasets, respectively. Columns $N\_S\ 10\%$ and $N\_S\ 99.9\%$ indicate the number of samples of that particular size needed to achieve the specified *coverage* value (10% or 99%). Numbers between brackets indicate that such number of samples would already achieve that *coverage*, however the number of samples is forced up to three. For the third classification context, imbalanced datasets preprocessed with SMOTE, the numbers of samples are derived from the original imbalanced datasets because the preprocessed datasets are already balanced. Tables containing specific value for each dataset, sample size and *coverage* value can be found on Appendix C.

		sizeOfMinClass		*maxSize*	
	% Min. Class [a]	*N_S* 10%	*N_S* 99.9%	*N_S* 10%	*N_S* 99.9%
Min.	0.08	3(1)	10	3(1)	3(1)
Max.	45.59	8	225	3	149
Mean	21.10	3.13	69.8	3	35.46
Standard deviation	16.41	0.35	75.93	0	41.92
Median	23.06	3	30	3	14

Table 3.2: Summary figures of subsample numbers for the 30 standard datasets.

		sizeOfMinClass		*maxSize*	
	% Min. Class [a]	*N_S* 10%	*N_S* 99.9%	*N_S* 10%	*N_S* 99.9%
Min.	0.77	3(1)	22	3(1)	9
Max.	35.51	27	1758	14	878
Mean	17.61	4.30	172.24	3.24	84
Standard deviation	11.70	4.30	313.07	1.84	156.68
Median	15.48	3	72	3	34

[a] Minority Class Distribution

Table 3.3: Summary figures of subsample numbers for the 33 imbalanced two-class datasets.

3.6 Results

In this section of the chapter the results of the contributions to the consolidation of decision tree algorithms are presented and discussed. The results encompass three different experiments.

The first experiment enables selecting one subsample type and coverage value as representatives for consolidated trees, and continue into the second experiment to compare the results of the different consolidated algorithms to the results published in the reference work which can be found in Appendix B. That work listed 16 genetics-based machine learning algorithms and classified them into three categories and five subcategories. It performed a hierarchical comparison by first determining the best algorithm for each subcategory, and then comparing these five algorithms to a set of classical algorithms for rule and tree induction (CART, AQ, CN2, C4.5, C4.5rules and RIPPER). These experiments were performed separately for each of the three classifications context independent of each other. The results of this experiment were published on [87].

In the second experiment presented in this chapter, the losers of the intra-subcategory experiment are not considered, and for each classification context, the winners of each subcategory, the consolidated algorithms, and classical algorithms are compared. Additionally, the classical algorithms used in the reference work, are complemented with the base algorithms of the new consolidated algorithms: CHAID*, C4.4 and CHAIC. It should be noted that the results for C4.5 present in the reference work, are the results using the C4.5 implementation from the KEEL platform. In this dissertation, the results given for C4.5 are for the implementation used to create the consolidated version of the algorithm. Even if the differences between both implementations of C4.5 are small, this implementation is more suitable to assess the benefits of consolidation. At the end of the experiment, the results of the three classification contexts are combined into a single comparison in order to find out the global performance of algorithms across all analyzed classification contexts. The results of this experiment were published on [86].

The third experiment compares the performance of base and consolidated decision trees from the point of view of the pruning strategy. The pruning process of decision trees aims to avoid overfitting to the training data by removing the most specialized branches. In presence of class imbalance, the branches identifying minority class examples could be these specialized branches. It is not uncommon for decision trees to be fully pruned trees down to the root node. Not only do these trees offer no explanation about the classification (they simply classify all examples as majority class), but they achieve a zero for most metrics used in the experiment. Thus, in the cases where the pruned decision trees just consist of a root node, the unpruned tree is used. This strategy is called the No Root Node (NRT) strategy. This experiment compares the strategies of always pruning, never pruning, using NRT, and also considering the probability estimation tree (PET) process as a pruning strategy, as the differences between regular and PET trees are mostly related to pruning. In this experiment, the same methodology as in the second experiment is used, but only focusing on C4.5, CHAID*, CTC45 and CTCHAID using different pruning strategies. The results of this experiment were also published on [86].

3.6.1 Results for different subsample types and coverage values (experiment one)

In this experiment the results of the CTC45 algorithm using two balanced subsample sizes and eleven coverage values are compared.

For each of the three classification contexts the results are analyzed from two points of view: the type of subsample used and the coverage value.

For the context of standard classification two performance measures are used, kappa and accuracy. Figure 3.2 shows the average kappa values for the 30 standard datasets, while Figure 3.3 shows the values for accuracy. Regardless of subsample type, both metrics show an uptrend when the coverage value is increased.

In order to compare the performance using different subsample sizes, the Wilcoxon Signed ranks test is used to make pairwise comparisons of the results for the same coverage value using different subsamples. Looking at the kappa and accuracy values, for every coverage value the average performance of CTC45 is better using *maxSize* subsamples. The results of the Wilcoxon statistical test on Table 3.4 show that for both metrics and all coverage values the performance of CTC45 is significantly better using the bigger samples, as expected.

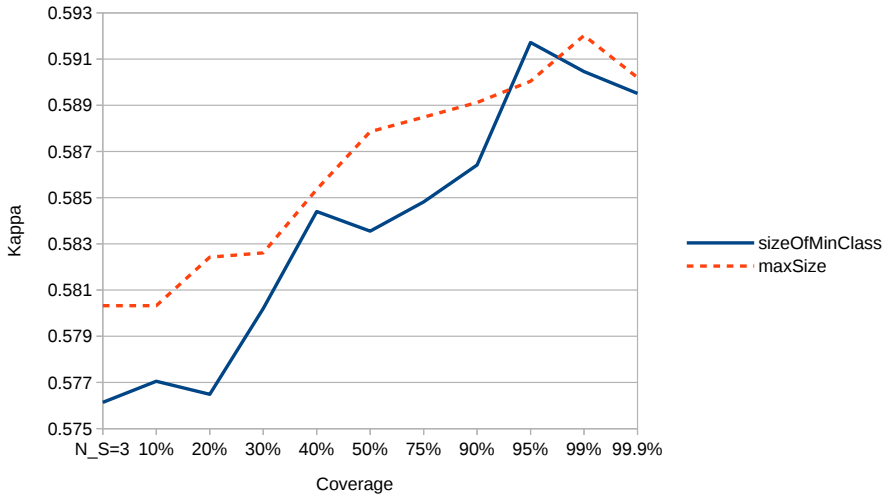

Figure 3.2: Performance of CTC45 for a range of coverage values with *sizeOfMinClass* and *maxSize* subsamples for standard datasets using kappa as the performance measure.

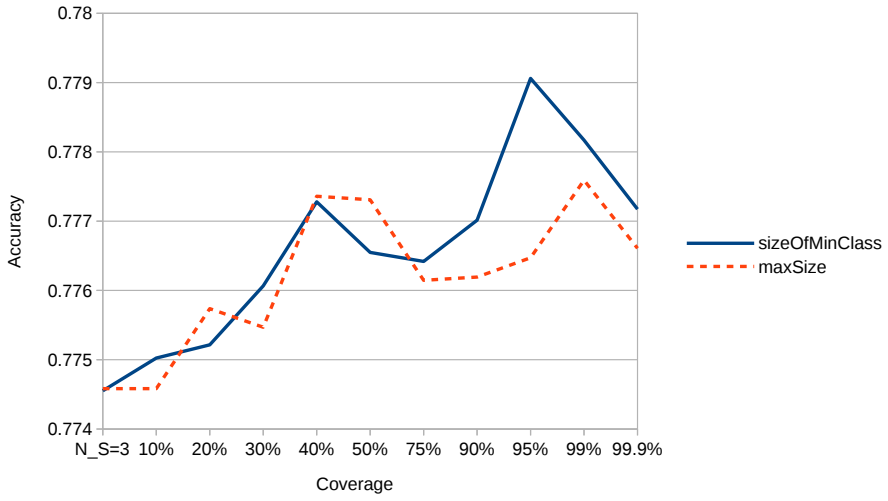

Figure 3.3: Performance of CTC45 for a range of coverage values with *sizeOfMinClass* and *maxSize* subsamples for standard datasets using accuracy as the performance measure.

Measure	*Coverage*	R^-	R^+	p-value	Hypothesis($\alpha = 0.05$)
kappa	*N_S*=3	40	**424**	0.0001	**Rejected in favor of *maxSize***
	10%	40	**424**	0.0001	**Rejected in favor of *maxSize***
	20%	33	**431**	0.00004	**Rejected in favor of *maxSize***
	30%	27	**437**	0.00002	**Rejected in favor of *maxSize***
	40%	45	**419**	0.0001	**Rejected in favor of *maxSize***
	50%	68	**396**	0.0007	**Rejected in favor of *maxSize***
	75%	89	**375**	0.0032	**Rejected in favor of *maxSize***
	90%	91	**374**	0.0036	**Rejected in favor of *maxSize***
	95%	111	**354**	0.0125	**Rejected in favor of *maxSize***
	99%	126	**339**	0.0285	**Rejected in favor of *maxSize***
	99.9%	128	**337**	0.0376	**Rejected in favor of *maxSize***
Accuracy	*N_S*=3	40	**424**	0.004	**Rejected in favor of *maxSize***
	10%	40	**424**	0.002	**Rejected in favor of *maxSize***
	20%	33	**431**	0.002	**Rejected in favor of *maxSize***
	30%	27	**437**	0.001	**Rejected in favor of *maxSize***
	40%	45	**419**	0.001	**Rejected in favor of *maxSize***
	50%	68	**396**	0.005	**Rejected in favor of *maxSize***
	75%	89	**375**	0.017	**Rejected in favor of *maxSize***
	90%	91	**374**	0.037	**Rejected in favor of *maxSize***
	95%	111	**354**	0.028	**Rejected in favor of *maxSize***
	99%	126	**339**	0.028	**Rejected in favor of *maxSize***
	99.9%	128	**337**	0.049	**Rejected in favor of *maxSize***

Table 3.4: Wilcoxon test comparing differences for kappa and accuracy for different subsample sizes over standard datasets.

For the context of two-class imbalanced datasets, the reference work only used one performance measure, the geometric mean between the true-positive and true-negative rates (GM). The results for this metric can be seen on Figure 3.4, and in this case, show a downtrend as the coverage value increases for both sample sizes. This is due to the true-positive rate decreasing at a higher rate than the true-negative rate increases as shown on Appendix E.

However, even if the reference work only used one metric for imbalanced datasets, other metrics that are well-suited for the class imbalance problem can be used, for example the F-Value. On Figure 3.5 it can be observed that in this case, the F-Value does increase with the coverage value, just like kappa and accuracy do on standard datasets. In fact, the increase on F-Value (4-6%) is noticeably greater than the decrease in GM (1.5-2%).

Similar to what happens with standard datasets, with imbalanced datasets the performance difference between using *sizeOfMinClass* and *maxSize* subsamples is also statistically significant. For the GM measure, the rank achieved by CTC45 using *maxSize* subsamples is always greater, and with significant differences most of the time, as seen on Table 3.5. Using the F-Value, the differences between subsamples are noticeably greater, with the Wilcoxon test always finding significant differences in favor of *maxSize* subsamples, as shown on Table 3.6.

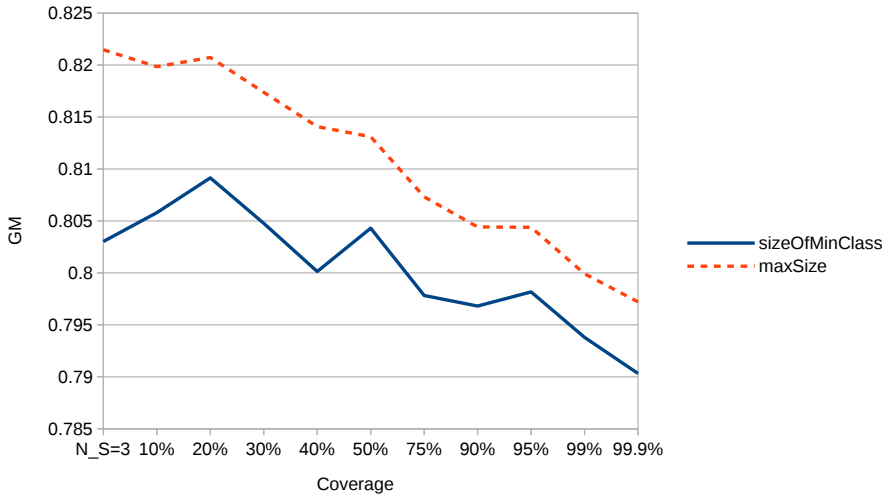

Figure 3.4: Performance of CTC45 for a range of coverage values with *sizeOfMinClass* and *maxSize* subsamples for imbalanced datasets using GM as the performance measure.

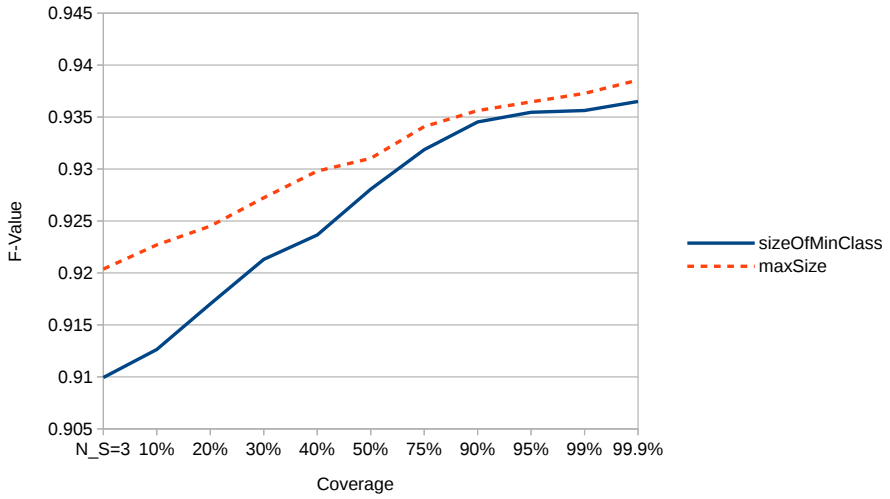

Figure 3.5: Performance of CTC45 for a range of coverage values with *sizeOfMinClass* and *maxSize* subsamples for imbalanced datasets using F-Value as the performance measure.

Measure	*Coverage*	R^-	R^+	p-value	Hypothesis($\alpha = 0.05$)
GM	*N_S*=3	121	**439**	0.004	**Rejected in favor of *maxSize***
	10%	134	**426**	0.009	**Rejected in favor of *maxSize***
	20%	146	**414**	0.016	**Rejected in favor of *maxSize***
	30%	114	**396**	0.037	**Rejected in favor of *maxSize***
	40%	165	**395**	0.039	**Rejected in favor of *maxSize***
	50%	174	**386**	0.057	Not rejected
	75%	155	**405**	0.025	**Rejected in favor of *maxSize***
	90%	171	**389**	0.05	**Rejected in favor of *maxSize***
	95%	169	**391**	0.046	**Rejected in favor of *maxSize***
	99%	203	**357**	0.166	Not rejected
	99.9%	148	**412**	0.018	**Rejected in favor of *maxSize***

Table 3.5: Wilcoxon test comparing differences for the GM for different subsample sizes over imbalanced datasets.

Measure	*Coverage*	R^-	R^+	p-value	Hypothesis($\alpha = 0.05$)
F-Value	*N_S*=3	39	**525**	0.00002	**Rejected in favor of *maxSize***
	10%	39	**525**	0.00002	**Rejected in favor of *maxSize***
	20%	43	**521**	0.00002	**Rejected in favor of *maxSize***
	30%	85	**476**	0.00048	**Rejected in favor of *maxSize***
	40%	79	**480**	0.00032	**Rejected in favor of *maxSize***
	50%	120	**441**	0.00413	**Rejected in favor of *maxSize***
	75%	64	**497**	0.00011	**Rejected in favor of *maxSize***
	90%	107	**454**	0.00193	**Rejected in favor of *maxSize***
	95%	164	**397**	0.03738	**Rejected in favor of *maxSize***
	99%	155	**406**	0.02493	**Rejected in favor of *maxSize***
	99.9%	103	**458**	0.00152	**Rejected in favor of *maxSize***

Table 3.6: Wilcoxon test comparing differences for the F-Value for different subsample sizes over imbalanced datasets.

Finally, for the context of imbalanced classification with datasets preprocessed with SMOTE, Figure 3.6 does not show that clear of a trend on GM when the coverage value increases. Using *maxSize* samples there is a slight downtrend with some zig-zags. On the other hand using *sizeOfMinClass* subsamples the up-and-downs on performance are noticeably greater, and while there is a downtrend starting from a coverage of 50%, a coverage of 99% still achieves better results than three subsamples. When using F-Value as the performance measure, however, as shown on Figure 3.7 there is a clear uptrend for both sample types when the coverage value increases.

When comparing the performance using different subsample sizes, the GM achieved by CTC45 using *maxSize* samples almost always surpasses the performance achieved using *sizeOfMinClass* samples. According to the Wilcoxon test results shown on Table 3.7 differences are not found to be significant although *maxSize* usually achieve a higher rank Using F-Value as the performance measure, *maxSize* samples achieve a better performance regardless of the coverage value, showing statistically significantly better performance for any coverage value as shown on Table 3.8.

It is worth noting that, if the results of CTC45 using *maxSize* subsamples and a coverage value of 99% are used, the differences between using CTC45 on imbalanced datasets and on the same datasets preprocessed with SMOTE, the results are statistically significant in favor of SMOTE+CTC45 as shown on Table 3.9. Until now, SMOTE was able to improve the average discriminating capacity of CTC45 classifiers, but never in a significant way.

Based on these results, *maxSize* subsamples combined with a high coverage value (coverage = 99%) are chosen as default values for consolidated trees. These values will be used in the experiments described in the following sections.

Figure 3.6: Performance of CTC45 for a range of coverage values with *sizeOfMinClass* and *maxSize* subsamples for imbalanced preprocessed with SMOTE datasets using GM as the performance measure.

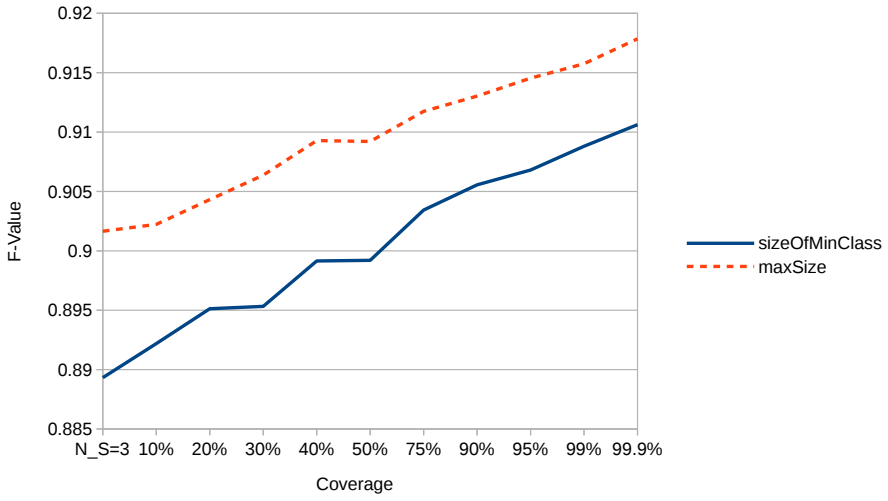

Figure 3.7: Performance of CTC45 for a range of coverage values with *sizeOfMinClass* and *maxSize* subsamples for imbalanced preprocessed with SMOTE datasets using F-Value as the performance measure.

Measure	*Coverage*	R^-	R^+	p-value	Hypothesis($\alpha = 0.05$)
GM	N_S=3	197	**364**	0.136	Not rejected
	10%	236	**325**	0.427	Not rejected
	20%	206	**355**	0.183	Not rejected
	30%	181	**380**	0.075	Not rejected
	40%	228	**333**	0.348	Not rejected
	50%	222	**339**	0.296	Not rejected
	75%	272	**289**	0.879	Not rejected
	90%	220	**341**	0.280	Not rejected
	95%	**310**	251	0.598	Not rejected
	99%	264	**297**	0.768	Not rejected
	99.9%	**361**	200	0.150	Not rejected

Table 3.7: Wilcoxon test comparing differences for the GM for different subsample sizes over imbalanced datasets preprocessed with SMOTE.

Measure	*Coverage*	R^-	R^+	p-value	Hypothesis($\alpha = 0.05$)
F-Values	N_S=3	126	**435**	0.0058	**Rejected in favor of *maxSize***
	10%	155	**406**	0.02493	**Rejected in favor of *maxSize***
	20%	108	**453**	0.0021	**Rejected in favor of *maxSize***
	30%	76	**485**	0.0003	**Rejected in favor of *maxSize***
	40%	128	**433**	0.0064	**Rejected in favor of *maxSize***
	50%	137	**424**	0.0.0103	**Rejected in favor of *maxSize***
	75%	168	**393**	0.0444	**Rejected in favor of *maxSize***
	90%	122	**439**	0.0046	**Rejected in favor of *maxSize***
	95%	123	**438**	0.0049	**Rejected in favor of *maxSize***
	99%	128	**433**	0.0064	**Rejected in favor of *maxSize***
	99.9%	144	**417**	0.0147	**Rejected in favor of *maxSize***

Table 3.8: Wilcoxon test comparing differences for the F-Value for different subsample sizes over imbalanced datasets preprocessed with SMOTE.

Measure	Comparison	R^+	R^-	p-value	Hypothesis($\alpha = 0.05$)
GM	CTC45 vs. SMOTE+CTC45	142	419	0.013005	**Rejected for SMOTE+CTC**

Table 3.9: Wilcoxon test comparing CTC45 with and without SMOTE on imbalanced data sets.

3.6.2 Results for consolidated tree construction algorithms against genetics-based and classical algorithms (experiment two)

This subsection describes the results of the second experiment. In this experiment the four consolidated algorithms (CTC45, CTC44, CTCHAID and CTCHAIC) are compared to their base algorithm and a wide set of genetics-based and classical algorithms.

The results for the statistical analysis of this experiment are found on Figure 3.8. In this Figure, the results of four comparisons are shown: two lines for the standard context, one using kappa as measure (1.Std-Kap) and another using the accuracy (1.Std-Acc), one line for imbalanced datasets (2.Imb-GM) and one line for imbalanced datasets using SMOTE (3.SMT-GM). Each of these lines shows the relative distances between algorithms according to their Friedman Aligned rank. The best algorithm for each comparison is the one on the left. Some of the left-most algorithms are covered by thick black lines. The algorithms covered by the thick line do not show statistically significant differences when compared to the best ranking algorithms according to Holm's test. An overview of all results for this experiment can be found on Table 3.10 and the full result tables are in Appendix F. For each experiment, algorithm, and used performance measure the Friedman Aligned rank is given, accompanied by the ranking position of each algorithm for a particular comparison. The first three positions of each ranking are highlighted. Some genetics-based algorithms do not have values for some experiments. As explained in Section 2.10 the GBML study first selected the best performing algorithms of each GBML subcategory for each classification context. In this experiment, for the context-by-context comparisons, the same algorithms are used. Algorithms that are not best-ranking for that experiment are not included in this statistical test and their cell in the table is empty. However for the global comparison, any GBML algorithm ranking best for at least one context is used.

As stated by the genetics-based machine learning (GBML) study [53], for the standard classification context the best GBML algorithms for each subcategory were XCS, GAssist, Oblique-DT, SIA and OCEC. Two performance measures are used in this context: the overall accuracy and the kappa statistic. In both cases CHAID* ranks first, above its consolidated version. The first six positions of the ranking stay almost the same for both performance measures, except for XCS and GAssist swapping the second and third positions. For the accuracy measure the Friedman Aligned ranks test computes a test statistic of 28.25 resulting in a p-value of 0.042 indicating the presence of statistically significant differences. Holm's test finds significant differences (adjusted p-value < 0.05) between CHAID* (the best ranking algorithm) and SIA, CART, RIPPER, CTC44, CTCHAIC, CN2, OCEC and AQ. For the kappa measure the Friedman Aligned ranks test also finds significant differences with a p-value of 0.041 (test statistic 28.34) and Holm's test finds differences between CHAID* and CART, OCEC, C4.5rules, CN2 and AQ. As these are $1 \times n$ tests, only differences with the control algorithm (the best ranking algorithm) are tested.

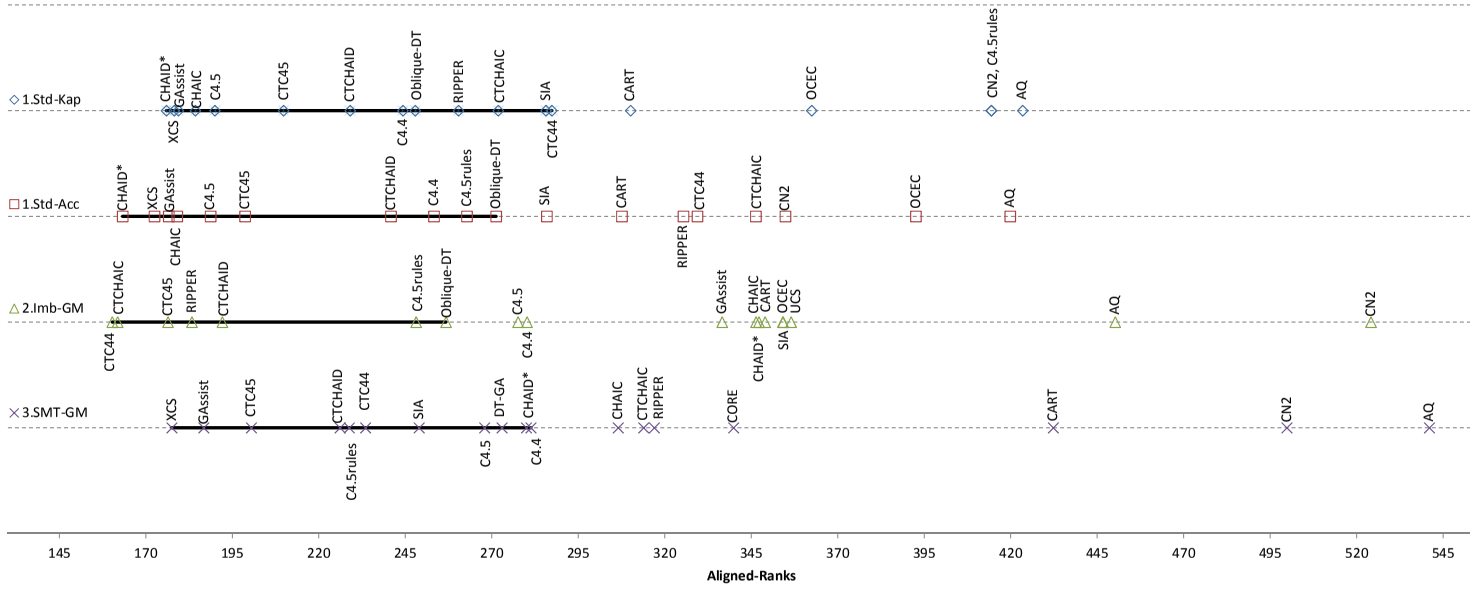

Figure 3.8: Visual representation of Friedman Aligned ranks for the three classification contexts.

		Standard				Imbalanced		Imb-SMOTE		Global	
		kappa		Accuracy		GM		GM			
	Algorithm	Avg. Rank		Avg. Rank		Avg. Rank		Avg. Rank		Avg. Rank	
GBML	CORE	—		—		—		339.89	(15)	1522.51	(19)
	DT-GA	—		—		—		272.95	(9)	967.13	(13)
	GAssist	179.40	(3)	176.62	(3)	336.59	(10)	186.74	(2)	801.26	(4)
	Oblique-DT	247.87	(9)	271.25	(10)	256.77	(7)	—		873.48	(9)
	OCEC	362.47	(15)	392.58	(17)	354.12	(14)	—		1283.78	(18)
	SIA	285.65	(12)	285.92	(11)	354.21	(15)	249	(7)	1004.81	(15)
	UCS	—		—		356.52	(16)	—		1154.17	(16)
	XCS	178.22	(2)	172.48	(2)	—		177.53	(1)	818.44	(6)
	AQ	423.47	(18)	419.87	(18)	450.21	(17)	541.05	(18)	1592.40	(20)
Classical	C4.5	189.97	(5)	188.7	(5)	277.58	(8)	267.92	(8)	811.13	(5)
	C4.5rules	414.4	(17)	262.8	(9)	248.12	(6)	228.80	(5)	1003.58	(14)
	CART	310.13	(14)	307.6	(12)	349	(13)	432.32	(16)	1212.48	(17)
	CN2	414.4	(16)	354.87	(16)	524.14	(18)	499.83	(17)	1632.14	(21)
	RIPPER	260.37	(10)	325.4	(13)	183.32	(4)	316.97	(14)	822.82	(7)
Added	CHAID*	175.95	(1)	163.3	(1)	347.23	(12)	280.03	(10)	897.79	(11)
	CHAIC	184.32	(4)	179.07	(4)	346.38	(11)	306.57	(12)	936.33	(12)
	C4.4	244.3	(8)	253.28	(8)	280.18	(9)	281.30	(11)	894.77	(10)
Consolidated	CTC44	287.37	(13)	329.43	(14)	160.29	(1)	233.53	(6)	751.97	(3)
	CTC45	209.8	(6)	198.68	(6)	176.39	(3)	200.52	(3)	649.46	(1)
	CTCHAIC	271.87	(11)	346.32	(15)	161.83	(2)	313.89	(13)	827.88	(8)
	CTCHAID	229.07	(7)	240.83	(7)	192.12	(5)	226.15	(4)	720.19	(2)

Table 3.10: Friedman Aligned ranks (and rank positions) for the three classification contexts and the global comparison.

Although CHAID*, a base algorithm, ranks first, the differences with its consolidated version, CTCHAID, are not found to be significant.

For the second context, imbalanced classification, the used performance metric is the GM. For this context the chosen genetics-based algorithms were UCS, GAssist, Oblique-DT, OCEC and SIA. In this case the best ranking algorithm is CTC44, followed by CTCHAIC and CTC45. The Friedman Aligned ranks test computes a test statistic of 30.87 (p-value 0.021) which indicates the presence of statistically significant differences. The Holm test finds statistically significant differences between CTC44 and C4.5, C4.4, GAssist, CHAIC, CHAID*, CART, SIA, OCEC, CART, AQ and CN2. In summary, four out of the best five ranking algorithms are consolidated, with the best ranking algorithm showing statistically significant differences with its base algorithm. All consolidated algorithms perform better than their base algorithm.

For the last context, where the imbalanced datasets are preprocessed with SMOTE, the GM is used as the performance measure. For this context the chosen genetics-based algorithms were XCS, GAssist, DT-GA, CORE and SIA. In this case the best ranking algorithm is XCS, followed by GAssist with CTC45 and CTCHAID ranking third and fourth, respectively. The test statistic is 30.38 and the p-value is 0.024. The algorithms performing significantly worse than XCS according to the Holm's test are CHAIC, CTCHAIC, RIPPER, CORE, CART, CN2 and AQ. In summary two out of the five best ranking algorithms are consolidated and only one consolidated algorithm, CTCHAIC, performs significantly worse than the best ranking algorithm. In this context, all consolidated

algorithms except CTCHAIC perform better than their base algorithm.

Finally, like in the previous experiment, a global analysis combining the results of the three contexts is performed in order to determine the robustness of algorithms across contexts. The rankings of this global analysis are found in Figure 3.9. While the classical algorithms and the consolidated algorithms are the same in the three previous contexts, some of the GBML algorithms only ranked for some of the contexts. For this comparison all GBML algorithms ranking for at least one context are used. For the standard dataset classification, only the kappa measure is used as using both measures would shift weight in favor of that particular context and kappa is better suited for contexts with class imbalance, which is the case for most datasets. In this case CTC45 ranks first followed by CTCHAID and CTC44, while CTCHAIC falls a little behind but without showing statistically significant differences with CTC45. The Friedman Aligned ranks test computes a p-value of 1.34×10^{-10}(test statistic 90.18) indicating a clear presence of statistically significant differences between the performance of the algorithms. According to Holm's test Oblique-DT, C4.4, DT-GA, SIA, C4.5rules, UCS, CART, OCEC, CORE, AQ and CN2 perform significantly worse than CTC45. As expected from the context by context results, all four consolidated algorithms perform better than their base decision tree algorithm as they show a more robust performance across contexts.

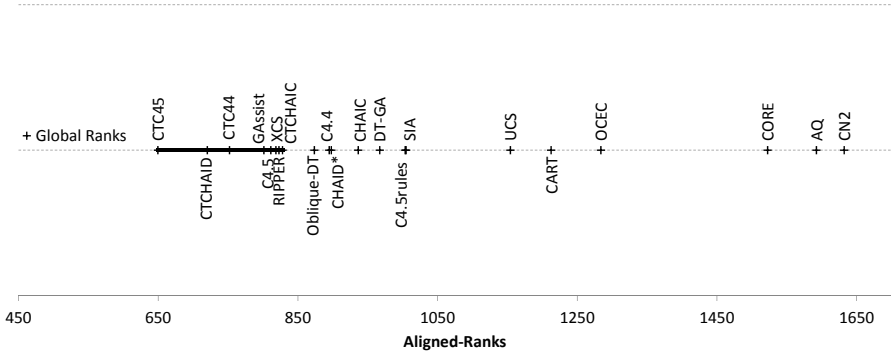

Figure 3.9: Visual representation of Friedman Aligned ranks for the three classification contexts.

3.6.3 Analysis of the effect of pruning on base and consolidated decision trees (experiment three)

As mentioned in the experimental setup, when pruning trees, especially in the presence of class imbalance, there is a risk that pruning will result in a tree being completely pruned to the root node. These trees simply represent the a priori probability of the sample used to train them. In the case of the simple decision trees this would be the original class distribution of the dataset,

and in the case of consolidated trees, as the subsamples used to build them are balanced, this would be an equal probability of $1/c$ for all classes, c being the number of classes. In any case, a root tree would classify all new examples as majority class. For two of the metrics (kappa and GM) used in this chapter, this would result in a value of 0. Specifically for the datasets used in this thesis, this mostly happens in the context of imbalanced two class datasets, as most of the times, the use of SMOTE to balance the classes prevents trees from pruning to the root.

In the presence of class imbalance, the process of pruning trees can be detrimental to the performance of decision trees. Thus, the following experiment compares the results different strategies have on base and consolidated decision trees. The considered strategies were always using pruned trees, always using unpruned trees and NRT. Also, as the new consolidated PET decision trees, CTC44 and CTCHAIC, mostly differ from CTC45 and CTCHAID in the replacement of pruning with a probability correction, for this comparison PET trees are considered a fourth possible pruning strategy.

Two independent comparisons are performed, one for C4.5-based algorithms and another for CHAID*-based algorithms. The full results of this experiment are found on Appendix G.

First the results of C4.5-based algorithms are compared. Table 3.11 shows the numeric values of the Friedman Aligned ranks (and the rank positions) for the comparisons of C4.5-based algorithms whereas Figure 3.10 shows these rankings graphically. In these diagrams, each point references a variant and can be identified by the tree type and pruning strategy. The tree type can either be DT for base decision trees, or CT for consolidated trees. The pruning strategy can be one of four possibilities: *unp* for always unpruned trees, *pru* for always pruned trees, *NRT* for the Not Root Tree strategy and *PET* for the variants using Laplace correction instead of pruning. Looking at the first context, standard dataset classification, C4.5 trees always pruning the trees (DT-*pru*) and the NRT strategy always achieve the same result, meaning NRT always selects the pruned tree. For the second context, imbalanced dataset classification, the best ranking strategy is using unpruned trees (*unp*), followed by PET trees, the NRT strategy and finally pruned trees (*pru*), performing significantly worse than unpruned trees. For the third context, SMOTE-preprocessed two-class dataset classification, the performance of the *pru* and NRT strategies is the same, regardless of tree type (base or consolidated). That is, as in the standard classification context, in these cases, pruning never results in a single tree and unpruned trees are never used. In the experiments made for Section 3.6.1, found on Appendix D, it is observed that when using lower coverage values, differences appear between the *pru* and NRT strategies. But as in these experiments, a coverage value of 99% is used, differences are non-existent. This shows that CTC45 follows the same pattern as C4.5 does [30]: pruning is detrimental when class imbalance is present but if SMOTE is used pruning seems to help the generalization capacity of the classifier. From a global point of view C4.5 and CTC45 show the same pattern seen on Figure 3.11: NRT ranks first, followed by the pruned strategy, with unpruned and PET consolidated trees performing

	Standard				Imbalanced		Imb-SMOTE		Global	
	kappa		Accuracy		GM		GM			
	Avg. Rank		Avg. Rank		Avg. Rank		Avg. Rank		Avg. Rank	
DT-pru	85.43	(1)	76.612	(1)	173.97	(8)	146.68	(5)	410.12	(6)
DT-NRT	85.43	(2)	76.62	(2)	167.65	(6)	146.68	(6)	403.28	(5)
DT-unp	112.47	(5)	116.25	(5)	153.70	(5)	159.17	(8)	424.6	(7)
DT-PET	135.57	(6)	132.01	(6)	171.83	(7)	158.05	(7)	466.93	(8)
CT-pru	94.82	(3)	88.18	(3)	135.61	(4)	98.52	(1)	331.46	(2)
CT-NRT	94.82	(4)	88.18	(4)	97.60	(3)	98.52	(2)	291.09	(1)
CT-unp	176.97	(7)	192.9	(7)	76.94	(1)	125.67	(3)	369.86	(3)
CT-PET	178.5	(8)	193.23	(8)	82.71	(2)	126.73	(4)	378.65	(4)

Table 3.11: Friedman Aligned ranks (and rank positions) for different pruning strategies of C4.5-based trees.

significantly worse than NRT trees.

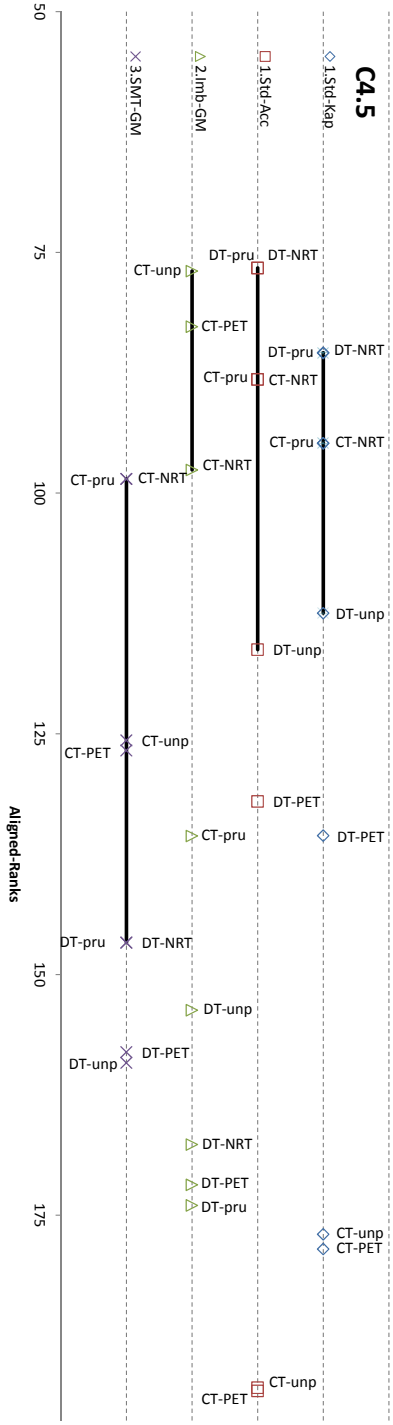

Figure 3.10: Context-wise Friedman Aligned ranks for pruning variants of C4.5-based algorithms.

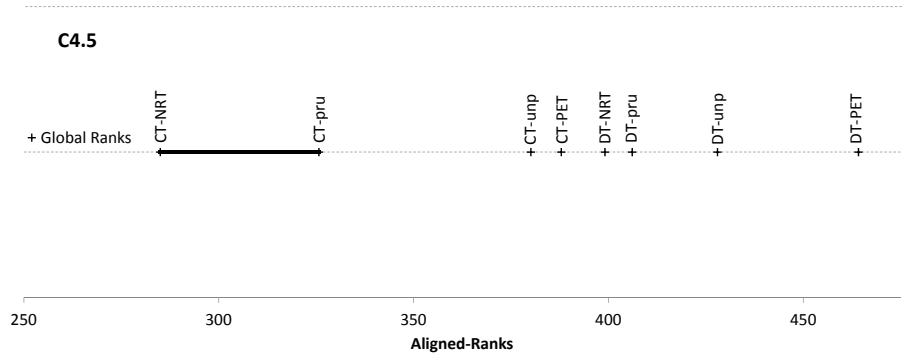

Figure 3.11: Global Friedman Aligned ranks for pruning variants of C4.5-based algorithms.

By looking at the results for CHAID*-based trees on Table 3.12 and Figure 3.12, the always pruning and NRT strategies do not match as much as with C4.5. Still, some patterns can be found. For standard dataset classification using the kappa as the measure base and consolidated trees follow the same pattern: pruned trees rank first, followed by trees using NRT strategy, trees that are never pruned and finally PET trees. Using accuracy as the measure no pattern can be found. NRT strategy works best for CHAID* and pruning for CTCHAID. For imbalanced datasets not pruning works best while always pruning ranks worst. Finally, for imbalanced datasets preprocessed with SMOTE variants always pruning rank best and ranks following the PET strategy rank worst. Looking at the global results on Figure 3.13 for consolidated trees never pruning works best, closely followed by the NRT strategy, and always pruning and the PET strategy work worse. For base decision trees, the NRT strategy works best, followed by never pruning, PET trees and finally always pruning. In both cases the best two strategies (switching first and second positions) are never pruning and NRT.

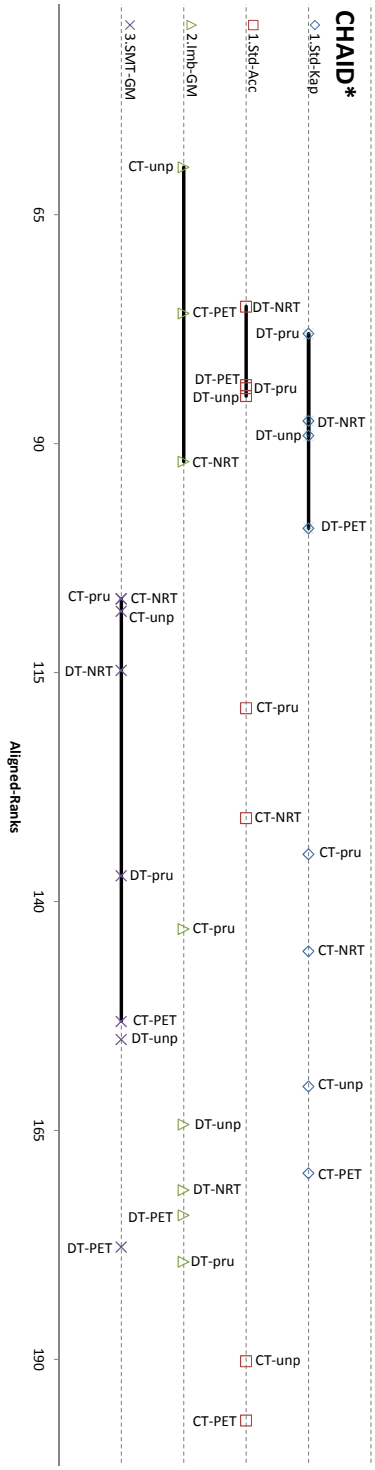

Figure 3.12: Context-wise Friedman Aligned ranks for pruning variants of CHAID*-based algorithms.

	Standard				Imbalanced		Imb-SMOTE		Global	
	kappa		Accuracy		GM		GM			
	Avg. Rank		Avg. Rank		Avg. Rank		Avg. Rank		Avg. Rank	
DT-pru	99.25	(1)	83.93	(3)	179.33	(8)	114.76	(4)	443.56	(8)
DT-NRT	77.98	(2)	75.03	(1)	171.52	(6)	137.18	(5)	386.86	(3)
DT-unp	87.52	(3)	84.8	(4)	164.35	(5)	153.09	(6)	403.33	(6)
DT-PET	89.12	(4)	83.6	(2)	174.26	(7)	155.06	(7)	417.82	(7)
CT-pru	145.43	(5)	118.88	(5)	143	(4)	106.94	(1)	387.49	(4)
CT-NRT	134.83	(6)	130.88	(6)	91.95	(3)	106.94	(2)	323.92	(2)
CT-unp	160.2	(7)	190.2	(7)	59.82	(1)	108.3	(3)	311.63	(1)
CT-PET	169.67	(8)	196.67	(8)	75.77	(2)	177.73	(8)	401.38	(5)

Table 3.12: Friedman Aligned ranks (and rank positions) for different pruning strategies of CHAID*-based trees.

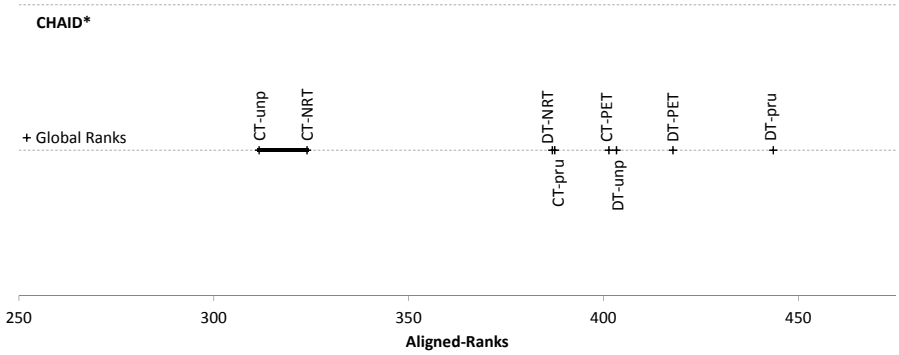

Figure 3.13: Global Friedman Aligned ranks for pruning variants of CHAID*-based algorithms.

3.7 Summary

Until recently the consolidation methodology was only applied to the C4.5 algorithm and used a range of fixed values for its two main parameters: the type of resampling and the number of subsamples used. While originally stratified samples that preserved the original class distribution were used, recent research suggests that balanced subsamples achieve the best results. However, changing the class distribution of the sample means that a subsample represents different classes to a different degree. With each sample, the minority classes are more represented than the majority classes. This requires a way to determine an adequate number of samples. As each dataset has a different class distribution, the number of samples is particular to each dataset.

This chapter proposes 'coverage-based resampling' as a solution this issue. The coverage is the probability of any example of the training sample being in at least one of the subsamples. Based on the minimum probability of any example being present in a single subsample, the number of samples required

to achieve a certain coverage is calculated.

This proposal is tested using two balanced subsample sizes and eleven coverage values over three classification context encompassing 96 datasets. These contexts are standard dataset classification, imbalanced classification, and imbalanced classification with datasets preprocessed with SMOTE. Comparing the subsample sizes, most of the time the bigger subsamples achieve a better average value for all metrics used. Also, whenever there are significant differences between the performance using different subsample sizes, it is in favor of the bigger subsamples. These subsamples are created by randomly undersampling all classes but the smallest one until they all have the same size as the minority class. This subsample type is referred to as *maxSize*. As for the different coverage values, for most of the performance metrics used, the value of the metric increases as the coverage value goes up. A value of 99% is selected for coverage in subsequent experiments.

With the goal of extending the consolidation methodology to algorithms other than C4.5, this chapter also proposes new algorithms to be consolidated. The already existing C4.4 algorithm is suggested as one of the replacements as base algorithms. New algorithms are also proposed. One of these algorithms is CHAID*. CHAID* is a variant of the widely-known CHAID algorithm, but unlike the original, CHAID* can handle continuous variables, and also performs pruning procedures like C4.5 does. These modifications allow it to be compared on the same problems as C4.5. Another one of the proposed algorithms is CHAIC. This algorithm proposes modifying CHAID* in the same manner that C4.5 was modified to create C4.4. Thus three new consolidated algorithms are proposed CTCHAID, CTC44 and CTCHAIC.

The results of the four base and four consolidated algorithms are compared to the results published by the reference work discussed in Section 2.10. The base and the consolidated tree algorithms are compared to a set of 16 genetics-based and five other classical rule and decision tree induction algorithms over the same three classification contexts used for the coverage experiment.

Results show that the four consolidated algorithms perform competitively. In summary, for two-class imbalanced dataset classification, four of the first five ranking (and also the top three) positions belong to consolidated algorithms. When SMOTE is applied to imbalanced two-class datasets almost all consolidated algorithms rank in the top half and above their respective base algorithms with the exception of CTCHAIC. For standard multi-class dataset classification, base decision trees rank better than their consolidated counterparts but without significant differences if kappa is used as a performance measure. The only significant differences present between the top ranking decision tree algorithm and two consolidated algorithms (CTC44 and CTCHAIC) appear when using the overall accuracy as the performance measure. This stands to reason as these two consolidated algorithms do not prune decision trees, while CHAID* trees are always pruned in this context, and the pruning procedure used, C4.5's pessimistic pruning, specifically aims to maximize accuracy in detriment of other measures. When comparing all algorithms globally, taking the results of the 96 datasets into account, CTC45 ranks first closely followed by the rest of consolidated al-

gorithms showing no statistically significant differences. All base decision tree algorithms rank lower than their consolidated counterpart. The global comparison show the robust performance of consolidated algorithms, especially CTC45 that are able to perform competitively across all analyzed classification contexts.

This chapter also proposed a new pruning strategy. It is not uncommon for decision trees to fully prune a tree in cases of severe class imbalance. These trees offer no explanation of how unseen instances are classified because all new examples are classified as majority class. As these trees are of no interest when comprehensibility is required, it is proposed that whenever pruning results in a single root node, the unpruned tree is used. This strategy is referred to the Not Root Tree (NRT) strategy.

As a final experiment, this chapter compares different pruning strategies. The newly proposed NRT strategy, always using pruned trees, always using unpruned trees, and the strategies applied by Probability Estimation Trees (PET) such as C4.4 or CHAIC. This experiment is performed separately for CHAID*-based trees, and C4.5-based trees. Results show NRT to be the recommended strategy in most cases, as the results are usually equal or better than using pruned trees, except for the imbalanced classification with no preprocessing, where unpruned and PET trees rank better, at the cost of a higher complexity, as unpruned trees, by definition, are equally complex or more complex than pruned trees. On a global comparison NRT places better for C4.5-based trees while unpruned trees rank better for CHAID*-based trees because unpruned CTCHAID trees distance themselves much more from the nearest competitor on the SMOTE-less two class classification context.

Chapter 4

Contributions to PART-like ruleset algorithms

4.1 Introduction

This chapter focuses on the contributions made during this thesis to PART-like ruleset algorithms.

As described in Section 2.7, the PART rule induction algorithm creates partial C4.5 decision trees, and extracts a rule from each tree. These partial trees are built by first splitting the root node and then attempting to split (processing) the processed leaf with the lowest entropy value. Then, among the recently created nodes, the one with the lowest entropy value is processed and so on, until a subtree is finished. Subtrees are finished when no more splits can be made, either because the current node does not have enough examples, potential children will not have enough examples, all possible splits have already been made, or no split has enough gain ratio. Once the subtree is finished, the "subtree replacement" process of C4.5's pruning algorithm is applied. If the replacement is successful and the subtree is replaced by a leaf, the algorithm continues, backtracking if necessary. If the replacement fails a subtree is considered stable and the partial tree is finished. Unlike in the standard C4.5 algorithm pruning is not applied after building the tree. A rule is extracted using the node with most weight, and examples not covered by this rule are used to repeat the process.

PART is one of the most widely used and cited rule induction algorithms. The paper contributing the algorithm [55], simply proposes the algorithm without suggesting or comparing any variations of it. This leads to asking if it would be possible to modify some of the decisions made during the algorithm creation to improve its performance. At the time of writing, according to Google Scholar, it has been cited 1486 times. However, since it was introduced, not many researchers have explored the possibilities of making such modifications to the algorithm.

First, as PART uses partial trees, making modifications to how the partial

tree is created results in potentially vastly different partial trees. If a decision tree is to be fully developed, the order in which the nodes are expanded does not really matter. The end result will be the same. In order to create a partial tree, on the other hand, a search algorithm has to be used. Changing the used search algorithm is the logical choice if the goal is to create different partial trees. Once the tree is built, PART only contemplates one way to select the leaf that will determine the rule to be extracted. Changing this criterion, while not affecting the partial tree building process at first, results in a different rule being selected, and in consequence, in different training sets for future iterations of partial trees.

Another possible alternative is the use of fully developed decision trees to create rules. This possibility eliminates the need of using a search algorithm to direct the decision tree building process as ultimately the entire tree is processed, and also eliminates the need of a criterion to select nodes suitable for rule extraction, as all nodes are processed.

Finally, algorithms other than C4.5 could be used to create the decision trees used to extract rules. This thesis already proposes some other decision tree algorithms in Chapter 3, namely, the CHAID* and CHAIC algorithms.

This chapter proposes new variants of the PART algorithm. First, four modifications are proposed to the original algorithm. From the resulting 16 variants, the best performing variant is selected, BFPART (Best-First PART). Another variant is proposed where instead of using partial decision trees, fully developed decision trees are used. This variant is called UnPART (as in not partial). Finally, CHAID*-based variants for PART, BFPART and UnPART are proposed.

The modifications proposed in this chapter do not only affect the discriminating capacity of the algorithm, but also the structural complexity of the generated classifiers and the computational cost to build them. Thus, the experiments in this chapter compare the performance of different algorithms from the perspectives of discriminating capacity, structural complexity and computational cost.

The structure of this chapter is as follows, Section 4.2, Section 4.3 and Section 4.4 describe the contributions presented in this chapter. Section 4.5 outlines the experimental setup for the experiments in this chapter, Section 4.6 analyzes the results of the experiment and, finally, Section 4.7 makes a brief summary.

4.2 The BFPART algorithm

As in all machine-learning algorithms, the decisions made during the classifier building process affect the performance. This section proposes modifying four of these criteria to improve PART's performance.

- *'Next Node to Develop' criterion:* The selection of the next node to be considered for development is crucial in the partial decision tree construction process. While in the standard C4.5 process, the nodes are analyzed

in pre-order, in PART, the node selected to continue developing the tree is always the child with the lowest entropy value. This node is more likely to result in a small subtree, and therefore, should produce a more general rule [55]. However, the selection is made from the child nodes obtained after the last split, without taking into account whether other nodes that were not analyzed at previous levels may be more suitable. It can thus be stated that PART uses the Hill Climbing algorithm (HC option) [132], to determine the next node to be analyzed. As with all local search algorithms, the Hill Climbing algorithm may not find a global minimum if there are local minima in the search space.

The Hill Climbing algorithm can be replaced with another better performing local search algorithm: the Best-First (BF) method [112, 132]. Global methods have shown to create better classifiers than local search methods. This method explores a graph by expanding the most promising node of the nodes that have not been explored, which are chosen according to a specified rule. In this context, the nodes of the partial decision tree currently under construction will be analyzed according to their entropy, and the node with the lowest value is selected, whatever its location in the tree.

This idea was used in the Best-First decision tree (BFTree) algorithm proposed in [141]. The author proposed two new pruning methods (based on pre- and post-pruning) with a pre-specified fixed size for the tree. As the node expansion order affects the final tree, and consequently the size of the tree, the depth-first order of Quinlan [128] could not be used in this case. These decision trees were evaluated in the context of boosting algorithms (no pruning methods) by Friedman et al in [57]. The authors found that building large trees decreases the error rate of each individual model, but it increases the error rate of the final model. Therefore, their aim was to find a set of trees to improve the accuracy while restricting the size of each individual tree to a predetermined size. Under these circumstances, it was necessary to define the node splitting order so that the best performing trees of the limited size were built. An implementation of this type of tree can be found in the WEKA site [75].

Two options for the *Next Node to Develop* criterion are proposed, namely Hill Climbing (HC) and Best-First (BF).

- *'Leaf for Next Rule' criterion:* After each of the partial decision tree has been built, the leaf node that determines the branch used to generate the next rule has to be selected. PART selects the leaf with the greatest weight, but considers only the nodes that are already labeled as a leaf in the partial decision tree, or the Treated Leaves (TL) option. These are the nodes for which after the analysis was done, the decision made was not to split the nodes (because all the instances belong to the same class or any other stop criteria is met); or, after the nodes had been divided, the "subtree replacement" operation of the pruning process was carried

out. However, there are some non-internal (structurally leaf-like) nodes that have not yet been analyzed, but which could be considered to build the rule as their "sibling" nodes. These leaf-like nodes may cover a greater number of instances. Thus, all of the nodes that are structurally leaf nodes could be analyzed (All Leaves, AL option).

Therefore, for the *Leaf for Next Rule* criterion, two options are proposed: Treated Leaves (TL) and All Leaves (AL).

- *'Pruning of Partial Trees':* The C4.5 algorithm, which is used by PART to develop the trees, prunes trees by default once they are built. This mechanism avoids overfitting. It ensures that trees are not too specific to the training sample and it also reduces the structural complexity. PART does not prune the partial trees once the building process stops. The possibility of pruning the partial trees should be considered, in order to determine which variants place better in the learning curve [78].

  For the *Pruning of Partial Trees* criterion, two options are considered: keeping partial trees unpruned (NP) and pruning partial trees (PR) using the same pruning strategy used by C4.5 described in 2.4.1.

- *'Priorization of Pure Nodes':* Pure nodes, which contain only examples from a single class, have the ideal entropy value of 0, and they are thus analyzed before their sibling nodes, and turned into leaves first. When all siblings are turned into leaves a subtree replacement is attempted. If the replacement fails, then the construction process stops. The early generation of leaf nodes in the process can lead to some branches of the tree not being developed because of a failed replacement. For this criterion, the option of the non-treatment of homogeneous nodes until the rest of the nodes have been analyzed is considered, (in the same way to how the PART algorithm treats small nodes) in order to further develop the partial trees and achieve a better discriminating capacity.

  For the *Priorization of Pure Nodes* criterion, two options are considered: Prioritize Pure Nodes (PP) and Do Not Prioritize Pure Nodes (DP).

Table 4.1 summarizes the criteria modified in the PART algorithm and the option proposals for each of them, and Table 4.2 presents the names of the 16 proposed variants in the last row.

Criterion	Original Option	New Proposal
Next Node to Develop	Hill Climbing (HC)	Best-First (BF)
Leaf for Next Rule	Treated Leaves (TL)	All Leaves (AL)
Pruning of Partial Trees	No Pruning (NP)	PRuning (PR)
Priorization of Pure Nodes	Prioritize Pures (PP)	Do not Prioritize (DP)

Table 4.1: Summary of the options explored for the generation of new variants of PART.

Hill Climbing							
Treated Leaves				All Leaves Leaves			
No pruning		Pruning		No Pruning		Pruning	
Prioritize Pures	Do not Prioritize	Prioritize Pures	Do not Prioritize	Prioritize Pures	Do not Prioritize	Prioritize Pures	Do not Prioritize
PART	HC_TL_NP_DP	HC_TL_PR_PP	HC_TL_PR_DP	HC_AL_NP_PP	HC_AL_NP_DP	HC_AL_PR_PP	HC_AL_PR_DP
Best-First							
Treated Leaves				All Leaves Leaves			
No pruning		Pruning		No Pruning		Pruning	
Prioritize Pures	Do not Prioritize	Prioritize Pures	Do not Prioritize	Prioritize Pures	Do not Prioritize	Prioritize Pures	Do not Prioritize
BF_TL_NP_PP	BF_TL_NP_DP	BF_TL_PR_PP	BF_TL_PR_DP	BF_AL_NP_PP	BF_AL_NP_DP	BF_AL_PR_PP	BF_AL_PR_DP

Table 4.2: Reference table for PART variant names.

An experiment was performed to determine the best among the 16 proposed PART variants. However, the methodology and datasets used for that particular experiment were different from the datasets and methodology primarily used in this thesis so the details about that experiment are placed on Appendix H. In summary, results showed that the BF_AL_PR_DP variant achieved the best results taking all metrics into account. This variant uses Best-First as the search algorithm when looking for the next node to develop (BF), takes all leaf-like nodes into account when extracting a rule (AL), prunes partial trees before extracting the rules (PR), and does not prioritize treating class-pure nodes (DP). This variant is exactly the opposite to the original PART algorithm regarding these characteristics. As this variant does not restrict the search to the nearest subtree, and postpones processing homogeneous nodes, partial decision trees are developed further than in the original PART. However, this is compensated by pruning the partial tree after the construction process and using selecting nodes in higher, undeveloped, levels of the tree to extract the rule. This variant was compared to the original PART, C4.5 and CHAID* using five performance metrics and was found to perform best among the compared algorithms. The results of that study can be found on Appendix H and were published on [85]. This variant was dubbed BFPART as it uses the Best-First search algorithm. The results presented later in this chapter, confirm the good results the chosen variants achieves using the common experimental methodology and datasets used by the reference work and this thesis.

Algorithm 4.1 shows BFPART's partial tree construction algorithm. The functions `new test, best, gain_ratio` and `average_gain_ratio` found in that algorithm work the same as they did in Algorithm 2.2. This algorithm also uses the `sort_by_entropy` and it works similarly to how it does in Algorithm 2.7 however in this case the newly created subsets are added to a global list and this global list is sorted. The `prune` functions in the same ways as it does in Algorithm 2.3. The main method of the BFPART algorithm is the same as PART's found on Algorithm 2.6, however, the `get_rule` method takes all leaves into account, not just treated leaves.

Input:
S: training data set
V: independent variables
$S_{global} = S$
Function `expand_BFPART(`S, V`):`
  mark S as treated
  $S_{global} = S_{global} \setminus S$
  if S *is empty* **then**
    turn node into leaf
    assign S same label as parent
    return
  end
  if S *meets stop criteria* **then**
    turn node into leaf
    assign S label of most common label of instances in S
    return
  end
  if *all cases on* S *are members of class* C_j **then**
    turn node into leaf
    assign label C_j to S
    return
  end
  foreach V_i *in* V **do**
    if V_i *is discrete* **then**
      $T_i =$ `new_test(`V_i`)`
    end
    if X_i *is continuous* **then**
      threshold = `best_split(`V_i`)`
      $T_i =$ `new_test(`V_i`)`
    end
  end
  $gr_{avg} =$ `average_gain_ratio(`T`)`
  $T_b =$ `best(`T`)`
  if *gain_ratio*$(T_b) < gr_{avg}$ **then**
    turn node into leaf
    assign S label of most common label of instances in S
    return
  end
  foreach *instance* I_j *in* E **do**
    foreach *outcome* O_i *in* T **do**
      if $O_i = I_{ji}$ **then**
        $S_i = S_i \cup I_j$
      end
    end
  end
  $S_{global} = S_{global} \cup E_i$
  `sort_by_entropy(`S_{global}`)` `// Exception: subsets with exactly 0 entropy are put last`
  foreach S_i *in* S_{global} **do**
    `expand_BFPART` (S_i,V)
    if *replacementFailed for* S_i **then**
      return
    end
  end
  `prune(`S`)`
  if S *is not leaf* **then**
    replacementFailed = true
  end
  return

Algorithm 4.1: BFPART's partial C4.5 tree construction algorithm.

4.3 The UnPART algorithm

The differences between PART and BFPART result in BFPART developing the partial trees further than PART does. PART uses a local search algorithm to guide the node processing order, and this search is restricted to the current subtree. BFPART uses the Best-First global search algorithm to look throughout the entire tree. If, for example, the first split in a tree creates three nodes and a stable subtree is found while developing the first of the subtrees, the other two are not even analyzed. BFPART can jump between different points of the decision tree and develop more branches. BFPART also uses the node's entropy value to select the next node to be developed, but unlike PART, makes an exception for class-pure nodes and does not treat them until later. These nodes have an entropy of zero. Delaying processing these nodes also delays the collapsing step, which in turn, delays the end of the tree building process, as the failure of replacing a subtree is the main stop criteria for partial trees. This could lead to a higher degree of overfitting the partial decision tree to the training data. However, the pruning used by BFPART's after the partial tree is built seems to mitigate this.

The results described on Appendix H show that BFPART creates classifiers with improved discriminating capacity, and significantly less structural complexity. So,

- What if instead of modifying the PART algorithm to create bigger but still partial decision trees, fully developed decision trees were used to extract rules?

The UnPART algorithm proposed in this chapter answers that question. The name itself is a play on words as UnPART (as in 'not PART') does not generate partial decision trees but still uses a very similar strategy to build rulesets based on decision trees.

The construction phase of decision trees in UnPART is the same as the base decision tree's construction process. The same stop criteria as the base tree construction process are used. There is no need to use a search algorithm to establish an order in which to develop nodes. All nodes are eventually developed using the decision tree algorithm's ordering method. There is also no need to discriminate between valid and invalid leaf-like nodes to extract the rules, as all leaf-like nodes are actually true leaf nodes. Like in BFPART, C4.5's subtree replacement pruning process is applied once the tree is built. Like in PART, the most populous node is used to extract a rule, the examples covered by the extracted rule are removed from the training set and the process is repeated until all training examples are covered by the ruleset. Algorithm 4.2 shows how UnPART creates decision trees by simply calling the base algorithm's construction process. In order to extract rules from decision trees it uses BFPART's

extraction procedure.

Input:
S: node whose biggest branch has to be identified
V: independet variables
Function `expand_UnPART(`*S,V*`)`**:**
    `expand_DecisionTree(`*S,V*`)`
    return

Algorithm 4.2: UnPART's tree construction algorithm.

4.4 CHAID*-based PART-like algorithms

PART uses the C4.5 decision tree induction algorithm as base algorithm to create the partial decision tree. BFPART and UnPART have originally been conceived with also C4.5 in mind as base algorithm.

However, the base tree algorithm can be replaced by other tree algorithms. The CHAID* algorithm, proposed in Chapter 3, is one such possibility. It is similar enough to C4.5 to enable easy integration into PART's ruleset creation process. Regarding PART, the two main differences between C4.5 and CHAID* are the criterion used to select the best split for a node and how splits by discrete features are handled. C4.5 uses the gain ratio of a split to determine its worth, and CHAID* uses the p-value of Pearson's chi-square test. When splitting a node using a discrete variable, C4.5 creates a branch for each possible value the variable can take, but CHAID* can group several values in the same branch.

From this point onward, in order to distinguish the same variant with different base, the _C45 (PART_C45, BFPART_C45 and UnPART_C45) and _CHD (PART_CHD, BFPART_CHD and UnPART_CHD) suffixes will be added when referring to C4.5-based and CHAID*-based algorithms, respectively.

C4.5-based PART and BFPART use the node's entropy to determine the node processing order. C4.5 uses entropy as part of the calculation to determine the information gain and gain ratio of possible splits, and select the best split. But CHAID* does not use entropy at any point of the process. CHAID* computes Pearson's Chi-square test for every possible split, and adjusts the p-value using the Bonferroni Correction. The split with the lowest adjusted p-value is used. Instead of using entropy, CHAID*-based PART and BFPART use the adjusted p-value of the best possible split for each node, selecting the node with the best split (lowest p-value). Entropy is not a value tied to a split, so even nodes that cannot be split have an entropy value. This is not the case of CHAID* nodes. But the p-value is tied to a split, so nodes that cannot be split, for example if all examples are already of the same class, do not have a p-value. In this cases, the p-value of the parent node is assigned to the node that cannot be split.

Table 4.3 summarizes the characteristics of each of the proposed variants compared to the original PART algorithm.

	C4.5-based			CHAID*-based		
	PART	BFPART	UnPART	PART	BFPART	UnPART
split criterion	gain ratio			p-value based on chi-squared test		
next node search	Hill Climbing	Best-First	-	Hill Climbing	Best-First	-
next node selection	entropy		-	p-value		-
discrete variables	one branch per value			grouped subsets		
missing values	spread between branches			combined with least significant branch if possible		
stopping criterion	failed replacement		-	failed replacement		-
leaf for rule	treated leaves	all leaves	all leaves are treated	treated leaves	all leaves	all leaves are treated
pruning	no	yes	yes	no	yes	yes

Table 4.3: Defining features of different PART-like algorithms.

4.5 Experimental Setup

The experiments in this chapter follow the same structure as the reference work and Chapter 3. The same train and test partitions for the same 96 datasets (described at the beginning of the contributions) are used.

The analysis of results in this section has been done in three stages, incrementally adding the number of compared algorithms. The first stage only compares the results of C4.5-based algorithms (three PART-like algorithms and C4.5 itself). The second stage compares the results of C4.5-based algorithms and their CHAID*-based equivalents. In the third stage these eight algorithms are compared to the set of genetics-based (GBML) and classical rule and decision tree induction algorithms used in the reference work and in Chapter 3.

The reference work only used performance metrics related to the discriminating capacity of the classifiers (kappa, overall accuracy, and GM). However, the modifications to the PART algorithm proposed in this chapter, greatly affect other aspects of the algorithm, such as the structural complexity of the classifiers and the computational cost required to build the classifiers. As there is no structural complexity and computational cost data for the results published by the reference work, the third stage is still limited to discriminating capacity metrics. However, on the first and second stages, structural complexity and computational cost metrics are used. For discriminating capacity, apart from kappa (first context) and GM (second and third contexts), the AUC is also used. For structural complexity two metrics are used: the number of rules in a ruleset (number of leaves in the case of C4.5 and CHAID*) and the *Length*, that as described in Section 2.8.1.1 is the average number of decisions in rules. For computational cost, the classifier construction time is used, measured in milliseconds. All of the experiments in this chapter were performed using the same hardware and software. The hardware node has an Intel Core i7-4790 processor (3.60 Ghz) and 16 GB of RAM. The operating system is Windows 7 Enterprise SP1, and all algorithms and variants were implemented in Visual C++.

The reference work divided the experimentation into three context: standard classification, imbalanced two-class classification and imbalanced two-class classification with datasets preprocessed with SMOTE. In order to keep the results section somewhat compact, context-by-context average results are given for every performance metric used, but the statistical tests for significance are performed using the 96 datasets together. This allows obtaining a global point

of view of the performance of each algorithm over different classification contexts. The full tables containing the values for each algorithm, dataset and performance metric combination can be found on Appendix I.

The reference work classified 16 genetics-based algorithms into five subcategories and an independent hierarchical study was performed for each classification context. First, an intra-subcategory study was performed, and then, the best algorithm of each subcategory and six classical algorithms were compared. As the third stage of this chapter compares the results of the algorithms proposed in this chapter and the reference work combining the results of the three classification contexts, any genetics-based algorithm winning at least one intra-subcategory comparison is considered. The GBML and all classical algorithms (AQ, C4.5, C4.5rules, CART, CN2, and RIPPER), plus CHAID*, which was not originally present, are compared to C4.5-based and CHAID*-based PART-like algorithms. For this third stage, the performance metrics are limited to those published for the GBML and classical algorithms in the reference work: kappa for the first group of datasets, the GM for the rest.

4.6 Results

As mentioned in the Experimental Setup, the results are divided into three parts. The first part compares the results of C4.5-based algorithms: UnPART_C45, BFPART_C45, PART_C45 and C4.5 itself. The second part adds CHAID*-based variants for the same algorithms, and compares the eight algorithms against each other. The final part compares these eight algorithms to results published in [53] for 16 genetics-based and five classical decision tree and rule induction algorithms, found in Appendix B. For simplicity, this chapter only includes the average values for every evaluation metric on each of the classification contexts, and the Friedman Average rank computed over the 96 datasets, in order to analyze the global performance of the algorithms on different classification contexts.

In this chapter, results are displayed in figures such as Figure 4.1. In these figures, each box represents an algorithm and its computed rank. A box with filled background denotes the best ranking algorithm for this set and metric, and a line between two algorithms denotes a lack of statistically significant differences ($\alpha = 0.05$) between those two algorithms according to the post hoc procedure. In this case, the number of compared algorithms allows to perform an efficient $n \times n$ pairwise analysis so the Bergman-Hommel procedure is used.

4.6.1 Results for C4.5-based PART-like algorithms

This subsection compares C4.5-based PART-like algorithms (PART_C45, BFPART_C45 and UnPART_C45) and C4.5 exclusively.

The results are divided into three groups according to the type of evaluation metric. First, the discriminating capacity metrics (Kappa, GM, and AUC) are discussed, followed by the structural complexity measures (*Number of Rules* and

Length), and finally, computational cost (classifier construction time measured in milliseconds).

At the end of the section, the results of different metrics are combined.

4.6.1.1 Results for discriminating capacity metrics

Table 4.4 shows the average values for the kappa and GM metrics for the four algorithms. For the standard datasets, the kappa statistic has been chosen as representative over the error rate, as class imbalance is also present in most standard datasets, and kappa is better suited in this case. Thus, Table 4.4 shows the average kappa values for standard datasets and the GM for imbalanced datasets.

	UnPART_C45	BFPART_C45	PART_C45	C4.5
1.Standard	0.6047	0.5989	0.6024	0.5851
(kappa)	(1)	(3)	(2)	(4)
2.Imbalanced	0.7294	0.7232	0.7105	0.7298
(GM)	(2)	(3)	(4)	(1)
3.Imb-SMOTE	0.8132	0.8121	0.8129	0.8068
(GM)	(1)	(3)	(2)	(4)
Friedman	166.03	193.08	213.40	197.49
Aligned ranks	(1)	(2)	(4)	(3)

Table 4.4: Average kappa and GM values for C4.5-based UnPART, BFPART, and PART, and C4.5.

For the standard datasets, UnPART achieves the best average value, followed by PART and BFPART, with C4.5 performing worst. This same order appears for the SMOTE-preprocessed datasets and the GM metric. On the other hand, for the original imbalanced datasets C4.5 achieves the best GM value, very closely followed by UnPART, with BFPART and PART lagging behind.

From a global point of view, according to the statistical tests for significance, statistically significant differences are found with a p-value of 0.006. UnPART ranks first, followed by BFPART and C4.5, with PART ranking worst. The Bergmann-Hommel post hoc procedure finds differences between the best- and worst-ranking algorithms, UnPART and PART, in favor of UnPART. These results are graphically shown on Figure 4.1.

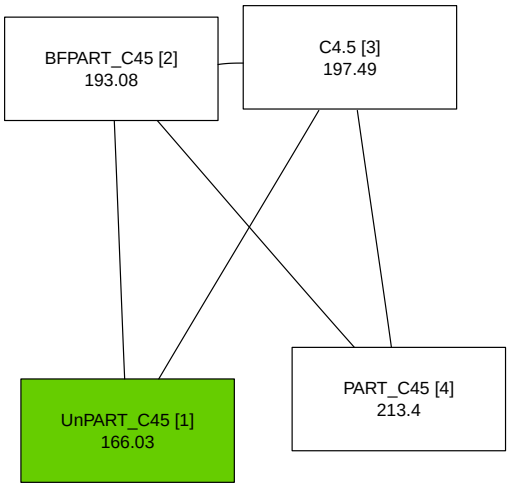

Figure 4.1: Friedman Aligned ranks and pairwise statistical differences for C4.5-based algorithms using the kappa and GM measures.

Table 4.5 shows the average results using the AUC measure. In this case, for each of the classification contexts, a different algorithm achieves the best average results: UnPART for standard datasets, PART for imbalanced datasets, and BFPART for preprocessed datasets. From a global point of view, PART performs best followed by UnPART, BFPART, and finally, C4.5. In this case statistically significant differences are also found by the Friedman Aligned ranks test (p-value=0.0002). The Bergmann-Hommel post hoc procedure concludes that C4.5 performs significantly worse than all of the other algorithms, whereas the other algorithms do not show significant differences among them, as seen on Figure 4.2.

In summary, for the kappa measure on standard datasets and the GM measure on SMOTE-preprocessed imbalanced datasets UnPART performs the best among the compared algorithms, whereas C4.5 performs best for non-preprocessed imbalanced datasets. Using these metrics, the Friedman Aligned ranks test and the Bergmann-Hommel post hoc procedure find UnPART to rank first and perform significantly better than PART. On the contrary, when AUC is used, PART performs best without significant differences to UnPART and BFPART, while C4.5 performs significantly worse than any of the rest. According to both statistical tests UnPART ranks in top positions.

	UnPART_C45	BFPART_C45	PART_C45	C4.5
1.Standard	0.8345	0.8301	0.8314	0.8134
	(1)	(3)	(2)	(4)
2.Imbalanced	0.8245	0.8268	0.8415	0.825
	(4)	(2)	(1)	(3)
3.Imb-SMOTE	0.8594	0.8602	0.857	0.8445
	(2)	(1)	(3)	(4)
Friedman	178.57	183.07	168.24	240.12
Aligned rank	(2)	(3)	(1)	(4)

Table 4.5: Average AUC values for C4.5-based UnPART, BFPART, and PART, and C4.5.

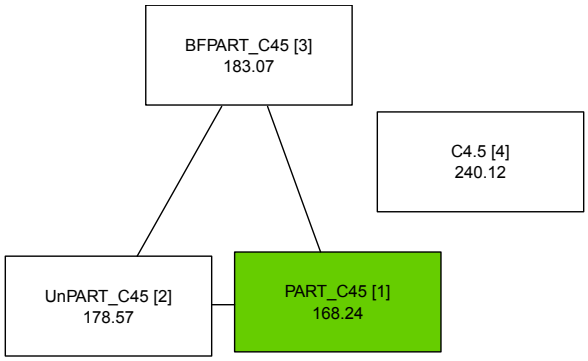

Figure 4.2: Friedman Aligned ranks and pairwise statistical differences for C4.5-based algorithms using the AUC measure.

4.6.1.2 Results for structural complexity metrics

This subsection compares the structural complexity of the algorithms.

Table 4.6 shows the average results for the *Number of Rules* metrics. Context-by-context average results are almost consistent, with UnPART and BFPART sharing the first positions, and PART and C4.5 placing third and fourth, respectively. Globally, as seen on Figure 4.3, the Friedman Aligned ranks test ranks UnPART first, followed by BFPART, PART and finally C4.5. PART generates considerably more rules on average compared to UnPART and BFPART, but less than C4.5. Both UnPART and BFPART show statistically significant differences when compared to PART and C4.5. PART, even though it creates significantly longer rules than UnPART and BFPART, still creates significantly simpler classifiers than C4.5 does, according to this metric.

	UnPART_C45	BFPART_C45	PART_C45	C4.5
1.Standard	13.25	13.15	26.55	36.44
	(2)	(1)	(3)	(4)
2.Imbalanced	5.27	5.39	6.22	11.27
	(1)	(2)	(3)	(4)
3.Imb-SMOTE	9.44	9.47	11.53	25.54
	(1)	(2)	(3)	(4)
Friedman	121.96	125.04	196.91	326.09
Aligned rank	(1)	(2)	(3)	(4)

Table 4.6: Average *Number of Rules* values for C4.5-based UnPART, BFPART, and PART, and C4.5.

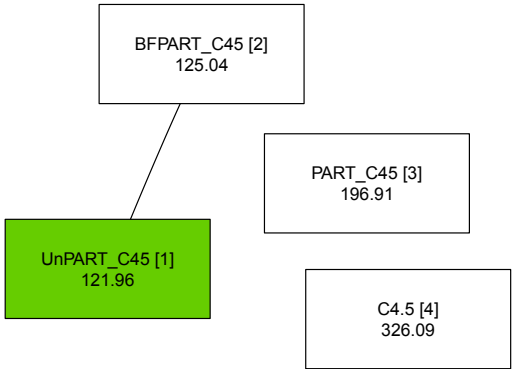

Figure 4.3: Friedman Aligned ranks and pairwise statistical differences for C4.5-based algorithms using the *Number of Rules* measure.

For the *Length* metric, according to the average results, the first position belongs to BFPART, UnPART or PART depending on the context. C4.5 places last for all three contexts, creating rules twice as long as the shortest ruleset. Average values are shown on Table 4.7. From a global point of view, according to the results of the statistical tests shown on Figure 4.4, UnPART ranks best, followed by BFPART and then PART. Finally, C4.5 creates significantly longer rules than the PART-like algorithms. It is not uncommon for algorithm orders to change between the average values and ranks, as average values tend to be skewed in presence of outliers. On the other hand, ranks smooth the effect of outliers.

	UnPART_C45	BFPART_C45	PART_C45	C4.5
1.Standard	2.35	2.31	2.46	4.88
	(2)	(1)	(3)	(4)
2.Imbalanced	1.40	1.46	1.52	3.48
	(1)	(2)	(3)	(4)
3.Imb-SMOTE	2.40	2.42	2.17	5.24
	(2)	(3)	(1)	(4)
Friedman	136.46	143.23	154.08	336.22
Aligned rank	(1)	(2)	(3)	(4)

Table 4.7: Average *Length* values for C4.5-based UnPART, BFPART, and PART, and C4.5.

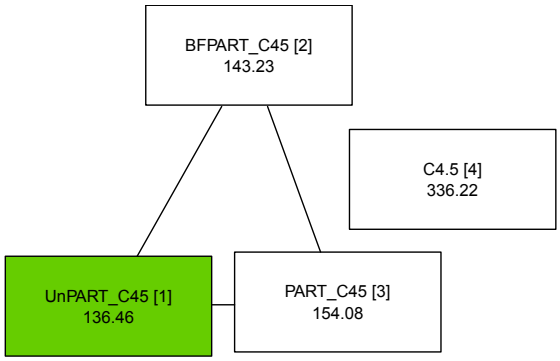

Figure 4.4: Friedman Aligned ranks and statistical differences for C4.5-based algorithms using the *Length* measure.

In summary, UnPART ranks best for both structural complexity metrics, showing statistical improvement over PART for the *Number of Rules* metric,

and C4.5 for both measures.

4.6.1.3 Results for computational cost

For the computational cost metric, the classifier construction time in milliseconds is used. As expected, C4.5 is the fastest, followed by PART, BFPART and finally UnPART, except for the standard classification context where BFPART and PART exchange places. Average values are shown on Table 4.8. If the median (not displayed in these results) is used instead of the average, the difference between algorithms is reduced from four-fold to two-fold. This indicates that some datasets act as outliers, greatly increasing execution time. The dataset-by-dataset results found in the additional material show that this happens with big datasets with mostly continuous variables. This makes sense as C4.5 finds the best split point by checking every possible value as a potential split point. This effect is augmented on datasets preprocessed with SMOTE, as this technique creates new artificial values for continuous variables.

According to the results shown on Figure 4.5, all algorithm pairs show significant differences.

In summary, C4.5 creates classifiers significantly faster than the rest of algorithms. This is expected, specially in the case of UnPART, as UnPART has to first create the same tree as C4.5 does, and then grow more trees. However, it should be noted that even though C4.5 works faster than the rule induction algorithms, it performs significantly worse for the AUC metric and creates significantly more complex classifiers.

	UnPART_C45	BFPART_C45	PART_C45	C4.5
1.Standard	252	229	237	75
	(4)	(2)	(3)	(1)
2.Imbalanced	103	91	88	62
	(4)	(3)	(2)	(1)
3.Imb-SMOTE	2335	1873	1465	521
	(4)	(3)	(2)	(1)
Friedman	271.80	230.95	175.96	91.29
Aligned rank	(4)	(3)	(2)	(1)

Table 4.8: Average Time values (in milliseconds) for C4.5-based UnPART, BFPART, and PART, and C4.5.

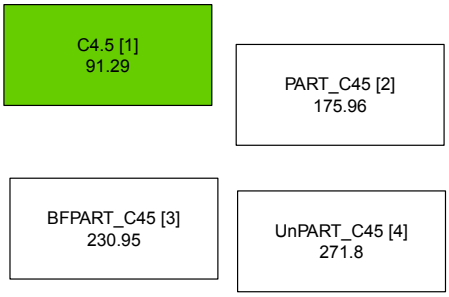

Figure 4.5: Friedman Aligned ranks and pairwise statistical differences for C4.5-based algorithms using the Time measure.

4.6.1.4 Global analysis of results

This last subsection focuses on comparing the four algorithms from a global point of view, combining the results of discriminating capacity, structural complexity, and cost metrics. This comparison gives twice the weight to the discriminating capacity (using kappa/GM and AUC) compared to structural complexity and computational cost measured in time. Since the main goal of a classifier is to generalize correctly, and results have shown that there are statistically significant differences among the algorithms in this regard, it makes sense to give more importance to this aspect of the classifiers. Regarding structural complexity, the rank orders are the same for both complexity measures, the common rank is used with the alias *Complexity* for the variable.

Table 4.9 shows the rank position each algorithm achieved for the measured metrics, and an average of rank positions is made.

	UnPART_C45	BFPART_C45	PART_C45	C4.5
kappa/GM	1	2	4	3
AUC	2	3	1	4
Complexity	1	2	3	4
Time	4	3	2	1
Average	2	2.5	2.5	3
	(1)	(2)	(3)	(4)

Table 4.9: Combining the results for all metrics for C4.5-based algorithms.

UnPART is determined to be the best among the compared algorithms. Even though it is the slowest to create classifiers, it creates the simplest models for both complexity measures and also ranks first for the combination of kappa

and GM. On the opposite side is C4.5: it is the fastest to create the models but creates the most complex and less accurate classifiers for AUC (second to last for the combination of kappa and GM). BFPART and PART fight for the middle positions. BFPART does not rank first or last for any of the measures, whereas PART performs worst for kappa and GM but best for AUC. From a global point of view BFPART ranks slightly above PART.

4.6.2 C4.5-based vs CHAID*-based PART-like algorithms

This subsection compares the results of C4.5-based algorithms and their equivalent CHAID*-based algorithms. The same performance measures as in the previous subsection (kappa, GM, AUC, *Number of Rules*, *Length* and Time) are used.

4.6.2.1 Results for discriminating capacity metrics

Table 4.10 shows the average values for the four C4.5-based and four CHAID*-based algorithms. Just as in Section 4.6.1, thus kappa is used for the standard datasets and the geometric mean (GM) for the rest. Results show that C4.5-based algorithms achieve the best results most of the time. UnPART_C45 achieves the best average results for standard and SMOTE-preprocessed imbalanced datasets followed by PART_C45 and BFPART_C45. The original C4.5 decision tree algorithm achieves the worst average results for these datasets. On the other hand, when the original imbalanced datasets are used C4.5 achieves the best GM, followed by the C4.5-based PART-like algorithms. There is more variance when the results of the CHAID*-based variant algorithms are observed, however, PART_CHD gets the best average positions among CHAID*-based algorithms.

	UnPART_C45	BFPART_C45	PART_C45	C4.5	UnPART_CHD	BFPART_CHD	PART_CHD	CHAID*
1.Standard	0.6047	0.5989	0.6024	0.5851	0.589	0.5871	0.5966	0.5906
(kappa)	(1)	(3)	(2)	(8)	(6)	(7)	(4)	(5)
2.Imbalanced	0.7294	0.7232	0.7105	0.7298	0.6797	0.681	0.6836	0.6793
(GM)	(2)	(3)	(4)	(1)	(7)	(6)	(5)	(8)
3.Imb-SMOTE	0.8132	0.8121	0.8129	0.8068	0.7948	0.785	0.7886	0.7851
(GM)	(1)	(3)	(2)	(4)	(5)	(8)	(6)	(7)
Friedman	298.78	314.46	359.24	342.70	444.06	441.56	427.88	447.31
Aligned rank	(1)	(2)	(4)	(3)	(7)	(6)	(5)	(8)

Table 4.10: Average kappa and GM values for the eight algorithms.

The Friedman Aligned ranks test finds statistically significant differences among the compared algorithms (test statistic = 47.51, p-value = $4.4e^{-}8$). Figure 4.6 shows the average ranks and the significant differences found by the Bergmann-Hommel post hoc procedure. UnPART_C45 ranks first for all three contexts. From the other algorithms, BFPART_C45 is the one with the most behavior. It places third for the three contexts but it ranks second for the global comparison because it does not fall positions for any context like all other algorithms do. According to the Friedman Aligned ranks test and the pairwise test

UnPART_C45 and BFPART_C45 perform significantly better than all CHAID*-based algorithms. No significant differences are found between C4.5-based variants, or between CHAID*-based variants.

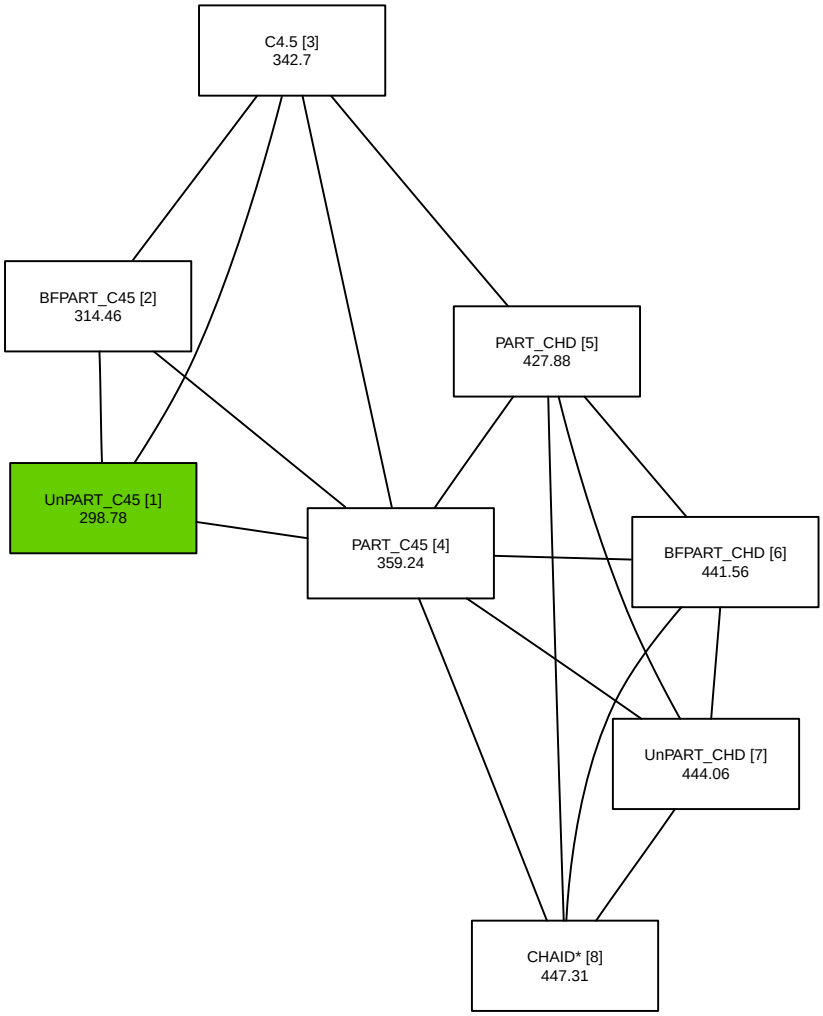

Figure 4.6: Friedman Aligned ranks and pairwise statistical differences for the eight algorithms using the kappa and GM measures.

When AUC is used as the discriminating measure, for standard datasets every CHAID*-based variant achieves a better average value than its C4.5-based counterpart except for UnPART. These values are shown on Table 4.11. For imbalanced datasets, C4.5-based algorithms perform better. The best average values are achieved by PART_CHD, PART_C45 and BFPART_C45, depending on the classification context.

	UnPART_C45	BFPART_C45	PART_C45	C4.5	UnPART_CHD	BFPART_CHD	PART_CHD	CHAID*
1.Standard	0.8345	0.8301	0.8314	0.8134	0.8319	0.8337	0.8425	0.8306
	(2)	(7)	(5)	(8)	(4)	(3)	(1)	(6)
2.Imbalanced	0.8245	0.8268	0.8415	0.825	0.8094	0.8109	0.8205	0.8094
	(4)	(2)	(1)	(3)	(8)	(6)	(5)	(7)
3.Imb-SMOTE	0.8594	0.8602	0.857	0.8445	0.8398	0.8392	0.8439	0.825
	(2)	(1)	(3)	(4)	(6)	(7)	(5)	(8)
Friedman	329.09	333.26	325.98	434.47	431.91	424.38	350.40	446.51
Aligned rank	(2)	(3)	(1)	(7)	(6)	(5)	(4)	(8)

Table 4.11: Average AUC values for the eight algorithms.

As shown on Figure 4.7, the Friedman Aligned ranks test and the Bergmann-Hommel post hoc procedure find statistically significant differences. PART_C45 ranks best followed by UnPART_C45. Both of these algorithms show significant differences compared to most CHAID*-based algorithms and C4.5. PART-like algorithms perform better than their base decision tree, with C4.5 and CHAID* achieving a worse rank than any ruleset algorithm.

In summary, depending on the used performance measure, UnPART_C45 or PART_C45 achieve the best position. For any of the measures used, both of these algorithm perform significantly better than most CHAID*-based algorithms and C4.5. In fact, when kappa and GM are used, UnPART_C45 performs significantly better than all CHAID*-based algorithms.

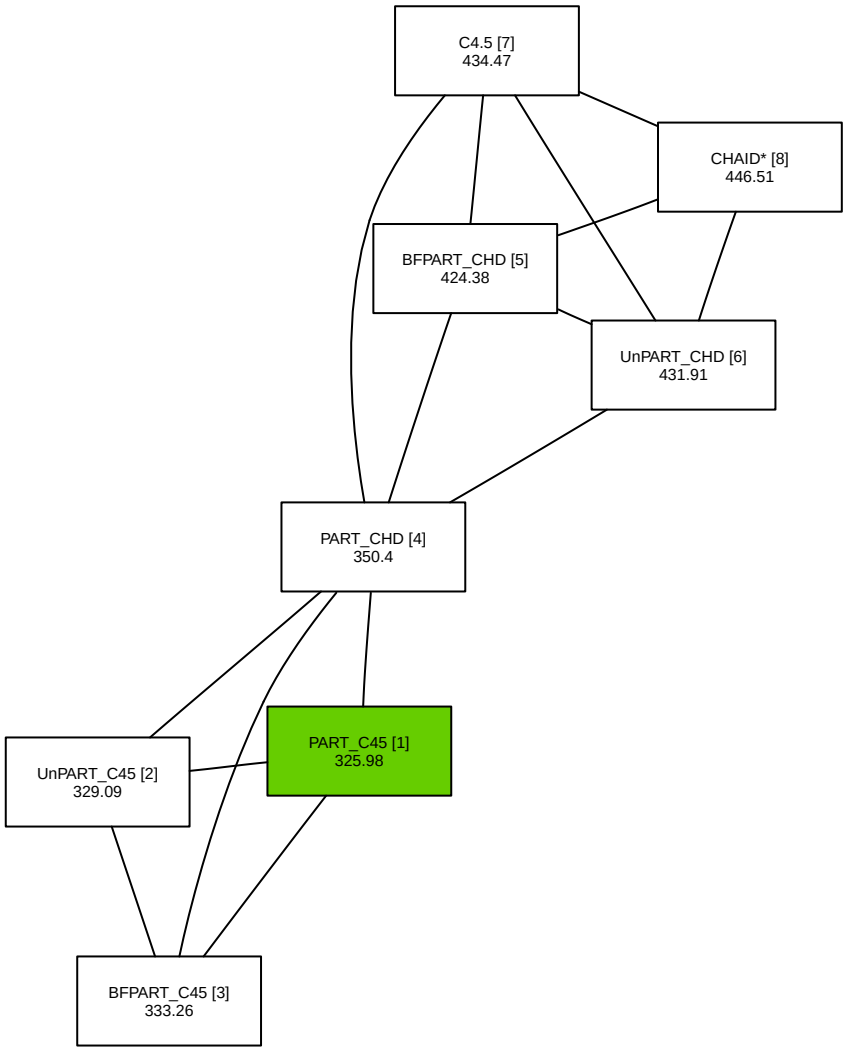

Figure 4.7: Friedman Aligned ranks and pairwise statistical differences for the eight algorithms using the AUC measure.

4.6.2.2 Results for structural complexity metrics

For the *Number of Rules* metric, in Table 4.12 it can be observed that CHAID*-based algorithms create rulesets with less rules than their C4.5-based equivalent. UnPART_CHD, BFPART_CHD and PART_CHD always achieve the lowest number of rules, whereas C4.5 creates the most, regardless of the classification context. From a global point of view, UnPART algorithms achieve the lowest rank, followed by BFPART and PART algorithms, and finally, the decision tree algorithms, CHAID* and C4.5, ranking worst.

The results of the analysis for statistical significance, depicted in Figure 4.8, show that UnPART_CHD achieves the least complex rulesets. The algorithms are grouped in four groups, where each group member does not show significant differences to other members of the same group. Closely following UnPART_CHD, BFPART_CHD and PART_CHD can be found without showing statistically significant differences with UnPART_CHD. The next group consists of BFPART_C45 and PART_C45, followed by another group made of PART_C45 and CHAID*. Finally, C4.5 stands on its own group, creating significantly more leaves than any other algorithm in the comparison.

	UnPART_C45	BFPART_C45	PART_C45	C4.5	UnPART_CHD	BFPART_CHD	PART_CHD	CHAID*
1.Standard	13.25	13.14	26.55	36.44	6.75	7.06	8.00	14.63
	(5)	(4)	(7)	(8)	(1)	(2)	(3)	(6)
2.Imbalanced	5.27	5.39	6.22	11.27	3.44	3.54	3.82	6.16
	(4)	(5)	(7)	(8)	(1)	(2)	(3)	(6)
3.Imb-SMOTE	9.44	9.47	11.53	25.54	6.52	6.81	7.84	17.78
	(4)	(5)	(6)	(8)	(1)	(2)	(3)	(7)
Friedman	355.11	360.43	514.47	673.43	216.48	223.80	241.76	490.52
Aligned rank	(4)	(5)	(7)	(8)	(1)	(2)	(3)	(6)

Table 4.12: Average *Number of Rules* values for the eight algorithms.

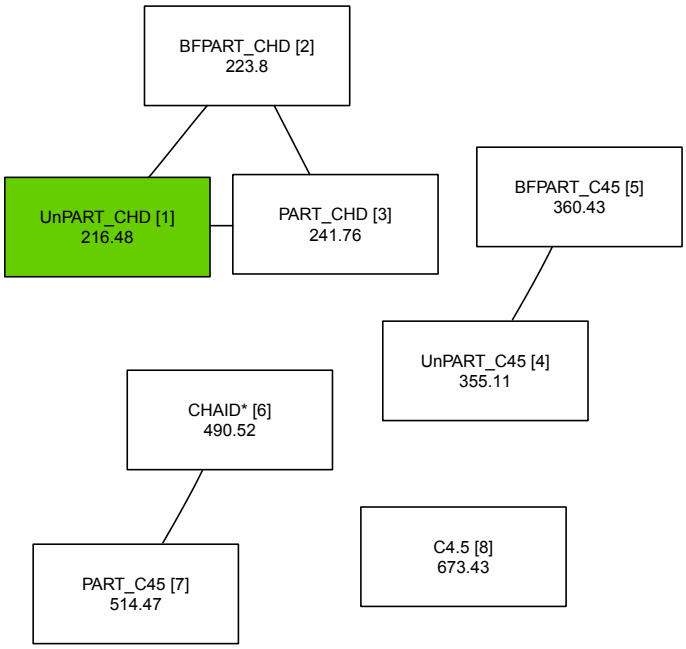

Figure 4.8: Friedman Aligned ranks and pairwise statistical differences for the eight algorithms using the *Number of Rules* measure.

For the *Length* metric, UnPART_CHD achieves the best average results for standard and imbalanced datasets, with PART_C45 achieving the best average result for preprocessed imbalanced datasets, as seen on Table 4.13. As it happens with the *Number of Rules* metric, decision tree algorithms achieve worse results than the PART-like algorithms.

The results of the statistical tests can be seen on Figure 4.9. Globally speaking, UnPART_CHD also ranks first for this measure. For this metric, just like the average results, ranks are also closer, only three algorithms generate significantly more longer rules than the rest: PART_CHD, C4.5 and CHAID*.

In summary, for both measures UnPART_CHD achieves the best average

	UnPART_C45	BFPART_C45	PART_C45	C4.5	UnPART_CHD	BFPART_CHD	PART_CHD	CHAID*
1.Standard	2.35	2.31	2.46	4.88	2.23	2.45	2.93	3.63
	(3)	(2)	(5)	(8)	(1)	(4)	(6)	(7)
2.Imbalanced	1.40	1.46	1.52	3.48	1.14	1.19	1.53	2.31
	(3)	(4)	(5)	(8)	(1)	(2)	(6)	(7)
3.Imb-SMOTE	2.40	2.42	2.17	5.24	2.39	2.58	3.02	4.27
	(3)	(4)	(1)	(8)	(2)	(5)	(6)	(7)
Friedman	275.52	280.47	300.34	691.09	234.40	282.61	412.02	599.55
Aligned rank	(2)	(3)	(5)	(8)	(1)	(4)	(6)	(7)

Table 4.13: Average *Length* values for the eight algorithms.

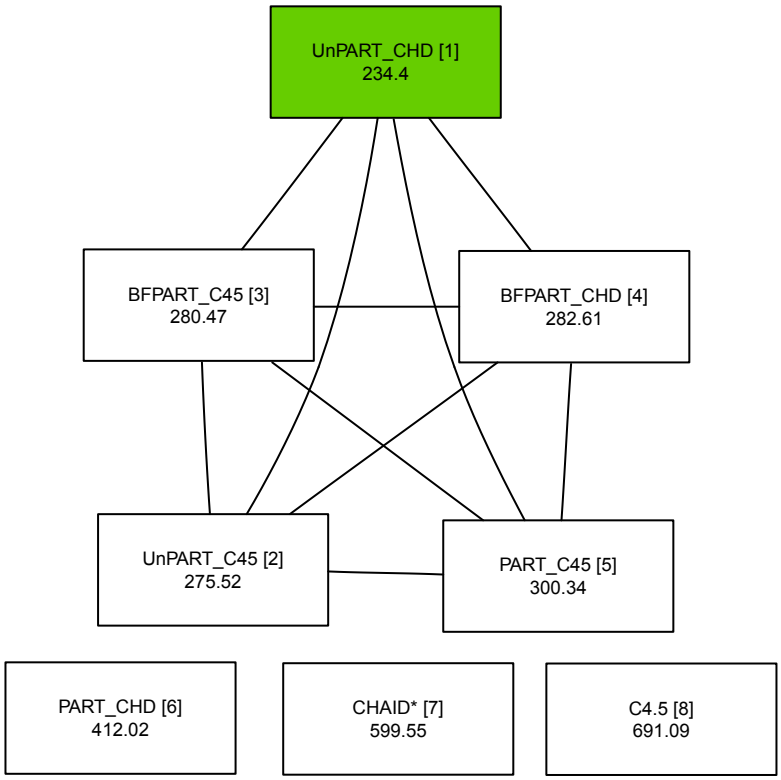

Figure 4.9: Friedman Aligned ranks and pairwise statistical differences for the eight algorithms using the *Length* measure.

results and ranks, closely followed by BFPART_CHD. Depending on the metric used, C4.5-based PART-like algorithms and PART_CHD also achieve similar results. However, independent of the performance metric used, decision tree algorithms create significantly more complex classifiers than their PART-like equivalents.

4.6.2.3 Results for computational cost metric

The results in Table 4.14 show the average construction times measured in milliseconds. As in Section 4.6.1, decision-tree algorithms are the fastest to create classifiers, followed by PART, BFPART and finally UnPART. In fact, this order is the most common, regardless of base decision tree or classification context. The only exceptions are BFPART_C45 and PART_C45 switching places when using standard datasets and PART_CHD being slower than UnPART_CHD and BFPART_CHD when using imbalanced datasets. Just as explained in Section 4.6.1, this is understandable as PART-like algorithms have to construct multiple decision trees. In the case of PART and BFPART, although the constructed trees are partial, the creation of multiple trees requires greater computational cost because each time a new partial tree is started, all of the examples not covered by previously-built nodes are packed together again, and node processing time is related to the number of examples in that node.

It is very noticeable that CHAID*-based algorithms require much more construction time than C4.5-based algorithms. The fastest CHAID*-based algorithm (CHAID* itself) being on average seven times slower than the slowest C4.5-based algorithm. Also, for discrete-valued attributes, CHAID* uses a heuristic algorithm to find the most significant grouping of values, whereas C4.5 simply creates one branch for each value, requiring no extra computational cost.

	UnPART_C45	BFPART_C45	PART_C45	C4.5	UnPART_CHD	BFPART_CHD	PART_CHD	CHAID*
1.Standard	253	229	237	75	1809	1767	1588	875
	(4)	(2)	(3)	(1)	(8)	(7)	(6)	(5)
2.Imbalanced	103	91	88	62	2294	2037	2641	1562
	(4)	(3)	(2)	(1)	(7)	(6)	(8)	(5)
3.Imb-SMOTE	2335	1873	1465	521	79596	76565	68174	17738
	(4)	(3)	(2)	(1)	(8)	(7)	(6)	(5)
Friedman	224.18	222.38	215.06	200.08	603.25	596.53	581.11	433.42
Aligned rank	(4)	(3)	(2)	(1)	(8)	(7)	(6)	(5)

Table 4.14: Average Time values (in milliseconds) for the eight algorithms.

According to the results of the statistical tests for significance, graphically shown on Figure 4.10, C4.5-based algorithms do not show significant differences among them when CHAID*-based algorithms are taken into account, with C4.5 ranking first. Next, CHAID* takes significantly longer than C4.5-based algorithms but significantly less than CHAID*-based PART-like ruleset algorithms.

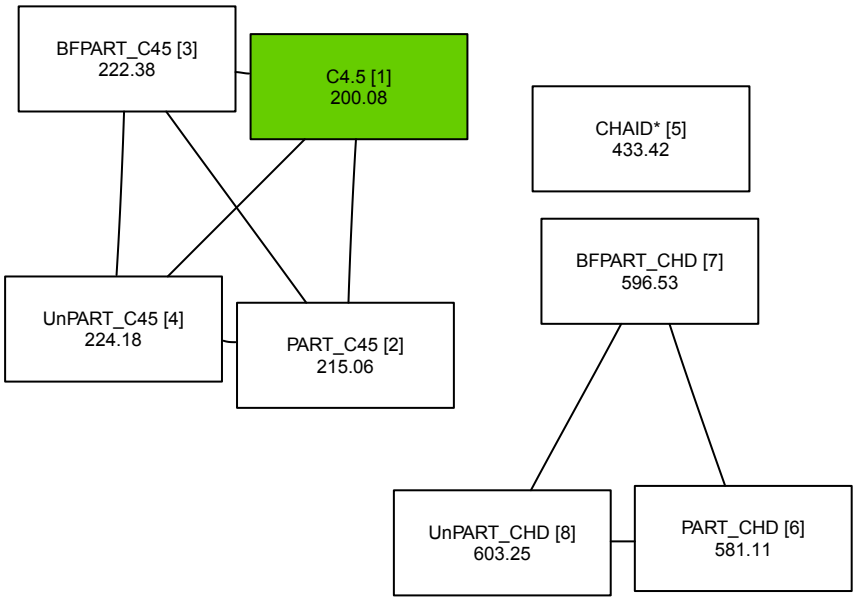

Figure 4.10: Friedman Aligned ranks and pairwise statistical differences for the eight algorithms using the Time measure.

4.6.2.4 Global analysis of results

Just like when comparing the C4.5-based algorithms, this last subsection compares the eight algorithms from a global point of view by combining the discriminating capacity, structural complexity and computational cost metrics. Table 4.15 shows the rank position each algorithm achieved for all metrics, and an average of rank positions.

Based on the average rank position UnPART_C45 is the best performing algorithm from a global point of view. UnPART_C45 creates some of the best classifiers without generating models that are too complex. Even though it is the slowest of all C4.5-based algorithms it is faster than any CHAID*-based algorithm.

For both algorithm bases (C4.5 and CHAID*), UnPART achieves the best average rank position and the decision trees achieve the worst position. Even though all C4.5-based PART like algorithms perform better than all CHAID*-based algorithms, CHAID*-based PART-like algorithms still perform better than C4.5.

Individually looking at the relationships between the base algorithm and the type of performance metric, a trade-off between discriminating capacity and complexity can be observed. C4.5-based algorithms create models with greater discriminating capacity but also greater complexity, whereas CHAID*-based

algorithms are worse classifying new examples but generate simpler models.

	C4.5-based				CHAID*-based			
	UnPART	BFPART	PART	C4.5	UnPART	BFPART	PART	CHAID*
kappa/GM	1	2	4	3	7	6	5	8
AUC	2	3	1	7	6	5	4	8
Number of Rules	4	5	7	8	1	2	3	6
Length	2	3	5	8	1	4	6	7
Time	4	3	2	1	8	7	6	5
Average	2.6	3.2	3.8	5.4	4.6	4.8	4.8	6.8
	(1)	(2)	(3)	(7)	(4)	(5.5)	(5.5)	(8)

Table 4.15: Combining the results for all metrics for C4.5-based and CHAID*-based algorithms.

4.6.3 PART-like algorithms vs GBML and classical algorithms

This subsection compares the results of PART-like and decision tree algorithms from Section 4.6.2 to results published in the reference work. In that work, 16 genetics-based rule induction algorithms were classified into three categories and five subcategories, and a hierarchical comparison was performed. The datasets were separated in the three context described at the beginning of this chapter, and the algorithms were compared for each classification context, independent of the other two. For each dataset group, first an intra-subcategory comparison was made, and the winner of each subcategory and other six classical non-evolutionary algorithms were compared.

From each evolutionary subcategory, any algorithm ranking first for at least one of the intra-subcategory comparisons is used. This results in a direct comparison of 21 algorithms.

In Section 4.6.1 and Section 4.6.2 structural complexity and computational cost metrics are used in conjunction with discriminating capacity metrics to evaluate the algorithms. However, in this section, results are restricted to the metrics used in the referenced work: kappa and accuracy for standard datasets, and GM for imbalanced datasets. For standard datasets kappa is select because the majority of standard datasets present class-imbalance, and kappa is a better suited measure in this case.

Like in previous comparisons, the Friedman Aligned ranks test is used to analyze the differences between algorithms for statistical significance. However, in this case it is not possible to show all pairwise significant differences (or lack of) because of the number of algorithms. So in this case, the used pairwise test is the $1 \times n$ Holm procedure. Results are shown on Table 4.16. For the standard classification context, the Friedman Aligned ranks test finds significant differences (test statistic = 237.91, p-value = $1e^{-}10$). In this case, due to the size of the algorithm set compared, the statistical analysis for pairwise differences

switches from a $n \times n$ test to a $1 \times n$ test, comparing the best ranking algorithm (one of this chapter's proposals) to the rest of the algorithms. UnPART_C45 ranks first and the post hoc pairwise test finds significant differences between it and SIA, CART, OCEC, CN2, C4.5rules and CORE. BFPART_C45, PART_C45 and PART_CHD rank the following 3 positions. For the imbalanced classification context, RIPPER ranks first according to the Friedman Aligned ranks test and the pairwise test finds significant differences with most of the algorithms. However, with the exception of PART_C45, every other C4.5-based algorithm does not perform significantly worse than RIPPER. For the imbalanced context preprocessed with SMOTE, XCS ranks first showing significant differences with about half of the algorithms. It should be noted that in this case RIPPER performs significantly worse than XCS for this context. From a global point of view, taking all three contexts into account, UnPART_C45 ranks first, followed by BFPART_C45. Significant differences are found with UCS, CART, OCEC, CORE, AQ, CN2.

UnPART_C45, BFPART_C45, C4.5 and Oblique-DT (ranked in this order) show the most robust behavior. These algorithms rank either in top positions in the three contexts, or their differences with the best ranking-algorithm are not found to be significant. RIPPER, best ranking algorithm for the second context, performs significantly worse than XCS in the third context. Also, XCS, best ranking algorithm of the third context performs significantly worse than RIPPER in the second context. UnPART_C45 on the other hand, as mentioned, ranks first for the standard classification context and it does not perform significantly worse than RIPPER and XCS in the other two contexts.

In summary, UnPART_C45 ranks first for the first context and from a global point of view, in both cases with BFPART_C45 following right after. Among all C4.5-based PART-like algorithms PART_C45 is the only to perform worse than C4.5. UnPART_CHD ranks above BFPART_CHD and its base decision tree, CHAID*. This ordering is similar to the one achieved in Section 4.6.2 only comparing PART-like algorithms and their base decision trees. However, this comparison allows seeing the competitive performance these algorithms have compared to a broader range of ruleset and tree induction algorithms. CHAID*-based PART-like algorithms sacrifice part of their discriminating capacity for significantly simpler models, which is not measured in this section. However, from a global point of view, none of them perform worse than UnPART_C45, the best ranking algorithm.

1.Standard			2.Imbalanced			3.Imb-SMOTE			4.Global		
Algorithm	Ranking	p-value	Algorithm	Ranking	p-value	Algorithm	Ranking	p-value	Algorithm	Ranking	p-value
UnPART_C45	197.3167		RIPPER	166.303		XCS	189.4091		UnPART_C45	732.95	
BFPART_C45	214.55	0.71	Oblique-DT	235.13	0.16	GAssist	203.9091	0.77	BFPART_C45	758.80	1
PART_C45	220.38	0.71	C45-Rules	236.9697	0.16	C45-Rules	247.3485	0.48	GAssist	761.49	1
PART_CHD	222.36	0.71	C4.5	259.13	0.16	BFPART_C45	262.7727	0.36	C4.5	781.56	1
GAssist	224.06	0.71	UnPART_C45	274.06	0.12	UnPART_C45	262.8939	0.36	RIPPER	799.08	1
XCS	225.58	0.71	BFPART_C45	279.77	0.09	SIA	263.4091	0.36	XCS	802.68	1
UnPART_CHD	233.23	0.71	DT-GA	309.65	0.02	PART_C45	268.0758	0.33	PART_C45	819.00	1
CHAID*	234.48	0.71	PART_C45	323.57	0.006	C4.5	291.5606	0.2	Oblique-DT	827.70	1
BFPART_CHD	237.33	0.71	GAssist	332.19	0.002	DT-GA	293.2424	0.2	DT-GA	919.86	0.22
C4.5	238.23	0.71	OCEC	347.12	0.0009	Oblique-DT	297.6667	0.2	PART_CHD	921.47	0.22
Oblique-DT	288.7	0.51	PART_CHD	359.83	0.0007	UnPART_CHD	325.1212	0.06	UnPART_CHD	932.86	0.17
UCS	293.53	0.40	SIA	362.18	0.0007	RIPPER	337.2424	0.03	BFPART_CHD	943.89	0.13
RIPPER	317.6	0.12	BFPART_CHD	363.62	0.0006	PART_CHD	344.7424	0.02	CHAID*	956.71	0.09
DT-GA	319.1	0.11	CART	366.53	0.0005	BFPART_CHD	347.9697	0.02	SIA	964.59	0.07
SIA	333.25	0.049	CHAID*	372.18	0.0003	CHAID*	351.1515	0.01	C4.5rules	969.01	0.07
CART	367.3	0.004	UnPART_CHD	372.8182	0.0003	CORE	364.6364	0.01	UCS	1136.97	0.000023
OCEC	423.93	0.00002	XCS	379.65	0.0003	CART	480.6212	∼0	CART	1201.41	∼0
CN2	474.4	∼0	AQ	479.45	0.0002	OCEC	488.9697	∼0	OCEC	1251.76	∼0
C45-Rules	474.4	∼0	CORE	513.59	∼0	UCS	489.4091	∼0	CORE	1492.21	∼0
AQ	489.5	∼0	CN2	590.78	∼0	CN2	564.6818	∼0	AQ	1578.82	∼0
CORE	596.33	∼0	UCS	632.42	∼0	AQ	612.1667	∼0	CN2	1625.63	∼0

Table 4.16: Friedman Aligned ranks and p-values for kappa and GM for all algorithms.

4.7 Summary

PART is one of the most widely known ruleset induction algorithms. Modifying some aspects of the rule induction process could improve the performance of the algorithm. However, since its inception not many variations have been proposed. This chapter proposes multiple variants of PART-like algorithms.

The first proposed variant differs from the original PART algorithm in four different aspects. First, it uses the Best-First global search algorithm to look for the next node to be developed. Hence the name, Best-First PART or BFPART. This variant does not process class-pure nodes until later than the original PART. Thus, the partial trees are developed further in BFPART. These partial trees are then pruned. Finally, any leaf node, processed or not, is considered a candidate to extract the rule.

Another variant uses the same strategy, sequentially creating decision trees and extracting a rule from each, but using fully developed decision trees instead of partial trees. This variant is named UnPART as the trees are not really partial.

Finally, CHAID*-based variants are proposed for PART, BFPART, and UnPART, bringing the total of proposed new algorithms to five.

Then, experiments have been carried out in three stages.

The first stage compares the three C4.5-based algorithms (PART_C45, BFPART_C45 and UnPART_C45) and C4.5 over the 96 datasets from the point of view of discriminating capacity (using kappa, GM and AUC), the structural complexity of the generated classifiers (*Number of Rules* and *Length*) and the computational cost to build the classifiers.

Results show that UnPART_C45 achieves the best discriminating capacity if kappa and GM are used as measures, performing significantly better than PART_C45, and second best discriminating capacity if AUC is used as the metric. For structural complexity, UnPART_C45 creates the simplest models. In fact, significantly simpler models than C4.5 (for both metrics) and PART_C45 (for one of the metrics). For computational cost C4.5 is significantly faster than any other algorithm but UnPART C4.5 is not significantly slower than PART_C45 or UnPART_C45.

In the second stage, the results of the first stage are compared to those achieved by the CHAID*-based variants of the same algorithms (CHAID*, PART_CHD, BFPART_CHD and UnPART_CHD).

According to the results of the second stage, UnPART_C45 and PART_C45 still create the classifiers with the best discriminating capacity. Significantly better than any CHAID*-based algorithm. For structural complexity, however, UnPART_CHD creates the simplest models for both complexity measures. UnPART_CHD rulesets contain significantly less rules than any C4.5-based algorithm. For execution time, any C4.5-based algorithm is significantly faster than any CHAID*-based algorithm.

The third stage, compares the results of the second stage to the results published by the reference work from the point of view of discriminating capacity. Thus, this stage compares 16 genetics-based, 7 classical (including CHAID*),

and 6 PART-like algorithms. All of them with explaining capacity.

The results of the third stage show that among the compared algorithms, UnPART_C45 ranks first, performing significantly better than six other algorithms. UnPART_C45 and BFART_C45 (ranking second) are two of the most robust algorithms. They are two of the only four (out of 21) algorithms to not perform significantly worse than the best ranking algorithm for any of the three contexts, and they hold the first two positions for the standard classification context. Both UnPART_C45 and BFPART_C45 rank better than their base decision tree algorithm, C4.5. All CHAID*-based PART-like algorithms rank better than CHAID* and none of them show significantly worse than UnPART_C45 from a global point of view. This, combined with their models being significantly simpler, with UnPART_CHD's models being simplest, make CHAID*-based PART-like algorithms a competitive alternative for instances where model complexity might be critical.

Chapter 5

Combining the results of consolidated and PART-like algorithms

In the previous chapters, Chapter 3 and Chapter 4, results are shown for independent experiments. In Chapter 3 the consolidation methodology is applied to decision tree algorithms and both the base decision tree algorithms and consolidated decision tree algorithms are compared against a wide set of decision tree and ruleset induction algorithms from the reference work. In Chapter 4 improvements are proposed to the PART ruleset induction algorithms by modifying the way the algorithms make certain key decisions. Eight variants of the PART algorithm are compared among themselves.

This chapter combines the results of Chapter 3 and Chapter 4 by comparing PART-like algorithms, base decision trees and consolidated decision tree algorithms to the genetics-based and classical algorithms from the reference work. Both Chapter 3 and Chapter 4 use the same datasets, same train and test splits for cross validation, and compare to the results published by the same reference work.

Chapter 3 compares different pruning strategies. For the results presented in this chapter, the best performing strategy has been chosen. Since C4.4 and CHAIC, and their consolidated versions, observed in Chapter 3, are considered pruning strategies, and are not the best performing for their respective algorithms, are not include in this chapter.

The results of this experiment can be found on Table 5.1. Just like in Chapter 3 this comparison can only be done from the point of view of discriminating capacity as model complexity and computational cost data is not available for the algorithms used in the reference work. The results are shown first context by context and them from a global point of view taking all contexts into account. For each comparison three columns are displayed. The first column shows the names of the algorithms, ordered by their Friedman Aligned rank,

from lower (best) to larger. The middle column shows the Friedman Aligned rank of each algorithm and the right-side column shows the p-value regarding the significance of the differences between the best ranking algorithm and the algorithm presented in that row.

For the standard classification context, the Friedman Aligned ranks tests finds the differences between algorithms to be significant. UnPART_C45 ranks first and the post hoc test finds significant differences against nine algorithms (RIPPER, DT-GA, SIA, CART, OCEC, CN2, C4.5rules, AQ, and CORE). UnPART_C45 is followed by BFPART_C45, CHAID* which are also contributions of this thesis, with XCS ranking fourth. For the imbalanced classification context CTC45 ranks first followed by RIPPER, CTCHAID and C4.5rules. The Friedman aligned ranks test and pairwise post hoc test find that most algorithms (15 out of 25) perform significantly worse than the best ranking algorithm, CTC45. In fact, some of the best ranking algorithms for the standard classification context (such as XCS) now perform significantly worse than CTC45. For the final context, imbalanced datasets preprocessed with SMOTE, XCS ranks best, showing statistically significant differences against 10 algorithms. XCS is followed by GAssist, CTC45 and CTCHAID.

From a global point of view, the Friedman Aligned ranks test finds significant differences (test statistic = 496.92, p-value=$1.78e^{-10}$), assigning the best rank to CTC45, whereas the post hoc tests finds differences between CTC45 and more than half of the algorithms (13 out of 25) including all CHAID*-based algorithms except CTCHAID, and most classical rule induction algorithms. Four of the top five positions are held by algorithms proposed in this thesis (CTC45, CTCHAID, UnPART_C45 and BFPART_C45). This shows the robust behavior of these algorithms as they rank in the top positions for at least two of the three contexts and when they are not on top, they do not perform significantly worse than the best ranking algorithm. GAssist is the fourth ranking algorithm from a global point of view but for the imbalanced classification context it performs significantly worse than CTC45 (p-value=0.004). A similar thing happens to XCS, it performs well for the standard classification context and ranks best for the imbalanced datasets preprocessed with SMOTE, but it performs significantly worse than CTC45 for imbalanced datasets without preprocessing, leading it to fall to the seventh position from a global point of view.

These results are consistent with the results seen on Chapter 3 and Chapter 4. Consolidated algorithms perform better than their base counterpart in most contexts and globally, and CTC45 and CTCHAID rank best. As for PART-like algorithms, their performance is different depending on the base algorithm, for C4.5-based variants UnPART ranks best, followed by BFPART and finally PART. For CHAID* it's the almost the opposite with CHAID* ranking best followed by PART, UnPART and BFPART. Chapter 4 showed that CHAID*-based UnPART, BFPART and PART rank better if the complexity of the models is taken into account because these variants sacrifice part of the discriminating capacity in favor of generating significantly simpler models. However, model complexity is not taken into account in this comparison.

In Summary, when combining the results of the contributions from Chapter

3 and Chapter 4 and comparing them to a wide set of algorithms from the reference work, the performed test show that contributions from this thesis perform best, with the top two positions of the ranking belonging to the consolidations of C4.5 and CHAID* when compared globally taking all contexts into account, closely followed by variations of the PART algorithm also proposed in this thesis.

1.Standard			2.Imbalanced			3.Imb-SMOTE			4.Global		
Algorithm	Ranking	p-value	Algorithm	Ranking	p-value	Algorithm	Ranking	p-value	Algorithm	Ranking	p-value
UnPART_C45	218.25		CTC45	194.3939		XCS	213.3333		CTC45	727.6354	
BFPART_C45	236.8833	0.72	RIPPER	197.6061	0.95	GAssist	231.1212	0.74	CTCHAID	800.6094	0.43
CHAID*	245.4833	0.72	CTCHAID	219.197	0.95	CTC45	248.0909	0.74	UnPART_C45	843.9583	0.41
XCS	247.25	0.72	C4.5rules	277.5758	0.37	CTCHAID	273.9697	0.74	GAssist	868.401	0.31
PART_CHD	247.5833	0.72	Oblique-DT	279.1667	0.35	C4.5rules	282.3485	0.6	BFPART_C45	874.6719	0.31
GAssist	247.7	0.72	C4.5	306.4242	0.19	SIA	300.4848	0.44	C4.5	893.6406	0.25
PART_C45	248.1833	0.72	UnPART_C45	322.7424	0.1	UnPART_C45	304.1818	0.44	XCS	898.6771	0.25
UnPART_CHD	257.2167	0.72	BFPART_C45	330.0606	0.07	BFPART_C45	305.4242	0.44	RIPPER	915.3229	0.21
BFPART_CHD	260.4833	0.72	DT-GA	359.5303	0.018	PART_C45	312.6667	0.33	PART_C45	943.2552	0.14
C4.5	265.6833	0.72	PART_C45	374.3939	0.008	C4.5	330.7576	0.21	Oblique-DT	955.8906	0.1
CTC45	294.1333	0.72	GAssist	382.7424	0.004	DT-GA	337.803	0.19	CHAID*	1000.1979	0.03
CTCHAID	312.6333	0.66	OCEC	395.303	0.002	CHAID*	340.4545	0.17	PART_CHD	1042.7552	0.007
Oblique-DT	324.0333	0.44	CHAID*	409.5606	0.0008	Oblique-DT	351.3939	0.11	DT-GA	1055.9167	0.004
UCS	327.1667	0.4	PART_CHD	410.0909	0.0008	UnPART_CHD	376.4242	0.03	UnPART_CHD	1061.2292	0.003
RIPPER	356.6	0.1	UCS	411.2121	0.0007	RIPPER	388.5152	0.02	BFPART_CHD	1069.6927	0.002
DT-GA	359.2	0.09	BFPART_CHD	415.303	0.0005	PART_CHD	390.5758	0.01	SIA	1096.0573	0.0009
SIA	367.8833	0.05	SIA	415.7273	0.0005	BFPART_CHD	396.7273	0.01	C4.5rules	1096.651	0.0009
CART	410.8	0.003	CART	417.3788	0.0004	CORE	412.0909	0.004	UCS	1277.7448	~0
OCEC	469.1333	0.00002	XCS	424.0152	0.0003	CART	539.6212	~0	CART	1352	~0
CN2	526.0667	~0	UnPART_CHD	426.5303	0.0003	OCEC	548.9394	~0	OCEC	1402.1875	~0
C4.5rules	526.0667	~0	AQ	539	~0	UCS	550.7727	~0	CORE	1664.474	~0
AQ	541.6667	~0	CORE	575.8333	~0	CN2	628.0152	~0	AQ	1756.0052	~0
CORE	656.4	~0	CN2	656.2121	~0	AQ	676.2879	~0	CN2	1806.526	~0

Table 5.1: Friedman Aligned ranks and p-values for kappa and GM for all algorithms.

Part IV

Conclusions

Chapter 6

Conclusions and Further Work

This chapter describes the conclusions extracted from the work presented in this thesis, and outlines potential further work based on said conclusions.

6.1 Conclusions

In some applications of classification, knowing the reasons why a classifier assigns an unseen example with one class or another is almost as important as the accuracy of the classification itself. This is specially true in fields where classification works as a decision support system and a human operator makes a final decision based on the outcome of the classifier, such as insurance fraud detection and medical diagnosis.

Some classification algorithms create models that are easily understandable by humans. Examples of such classifiers are decision trees and rulesets. These models are comprehensible because they classify unseen examples by applying simple decisions based on the values of the example's features. Other supervised learning algorithms such as Artificial Neural Networks and Support Vector Machines create complex models that do not offer a simple explanations of how classifications are made. These models work like black boxes; the output for a specific input is known but not how that output has been obtained.

This thesis aims to contribute to the field of comprehensible supervised learning algorithms. The contributions are grouped into the two aforementioned algorithm types: contributions to the consolidation of decision tree algorithms and contributions to PART-like ruleset induction algorithms. All of the experiments in the thesis follow a similar methodology, using the same 96 datasets and the same 5-fold cross validation partitions. The datasets are divided into three contexts, standard mostly multi-class dataset classification, imbalanced two-class classification and imbalanced two-class classification preprocessed with SMOTE (Synthetic Minority Over-sampling Technique). This methodology was origi-

nally used by the reference work for this thesis [64], that originally proposed a taxonomy for genetics-based algorithms for rule and tree induction and compared the performance of these algorithms against other classical algorithms for rule and tree induction.

6.1.1 Conclusions for the consolidation of decision tree algorithms

The most common way found in the literature of improving the results of decision tree algorithms is to create ensemble classifiers. These ensemble classifiers combine the outcome of multiple individual (usually dozens) classifiers each trained with a different sample, usually created by resampling the original training sample. In consequence, ensemble classifiers are not considered comprehensible. The ALDAPA research group developed a methodology that applies the ensemble process during the decision tree building process. This methodology is called consolidation and also uses multiple samples, but applies the ensemble's voting process on each node of the tree, by using the majority vote obtained by 'consulting' each of the samples. So, even if this methodology uses multiple samples, the outcome is a single decision tree, a consolidated decision tree. Recently the term 'Inner Ensembles' has been coined to refer to methodologies that work this way. The consolidation methodology was originally applied to C4.5 and the resulting decision tree algorithm was named CTC (Consolidated Tree Construction), although in this thesis it is referred to as CTC45 to distinguish it from other consolidated algorithms. CTC45 creates classifiers with a better discriminating capacity and more stability than C4.5 does. The conclusions of this thesis regarding the consolidation of decision tree algorithms can be organized in three groups: conclusions about the contributions to the creation of samples for consolidated decision trees, about the contributions to extending the number of consolidated algorithms, and about the contributions to the effect of pruning on decision trees and their consolidated version.

6.1.1.1 Conclusions about the Contributions to the Creation of Samples for Consolidated Decision Trees

Consolidation originally used samples that kept the dataset's original class distribution (stratified samples), and needed two parameters, the subsample size relative to the full sample size, and the number of samples (N_S). However, recent research suggests that CTC45 benefits from changing the class distribution. Specifically, creating subsamples with a balanced class distribution. However, this required a new way to create the subsamples, as the original resampling strategy did not take into account a particular issue of changing the class distribution: the fact that each sample does not represent all classes equally. The first contribution of this thesis to the consolidation methodology is the *coverage-based resampling*. This resampling technique takes into account the notion of *coverage*: the probability of all examples of the training sample being represented by the set of randomly created subsamples. Using a *coverage* value, a

distinct *N_S* value is computed for each dataset, removing the need to use a specific value for that parameter. The first experiment presented in the thesis analyzes the discriminating capacity of the CTC45 algorithm using two different balanced subsample sizes and several *coverage* values between 10% and 99.9%. The results of the tests for statistical significance show that CTC45 works significantly better using the bigger subsample sizes. These subsamples are achieved by keeping the full minority class and randomly undersampling every other class until the sample is balanced. This is the biggest possible balanced subsample without oversampling the minority class and is named *maxSize* in the experiments. Regarding the *coverage* value, most of the used performance metrics (kappa, accuracy, the geometrical mean, and F-Value) show an improvement when the *coverage* value is increased. In conclusion, using *maxSize* samples with a *coverage* value of 99% is considered the optimal configuration for CTC. This configuration offers a trade-off between using a fixed number of samples and the performance of the algorithm, and is the standard configuration used throughout the rest of this thesis. Using these parameters, comparing the performance of CTC45 with the original imbalanced datasets and the preprocessed datasets, there are statistically significant differences in favor of using SMOTE prior to CTC. When comparing results globally, taking all contexts into account, CTC45 has the most robust performance compared to the other 20 algorithms. It is able to achieve top ranks in all three contexts, not being hindered by the differences in the real-world problems for each context.

6.1.1.2 Conclusions about the Contributions to extending the Number of Consolidated Algorithms

Before the work on this thesis begun, the only consolidated algorithm was C4.5. Some contributions to consolidation focus on extending the consolidation methodology to other decision tree algorithms to evaluate if consolidation can improve the results of algorithms other than C4.5. For this, one existing decision tree algorithm, C4.4, and two novel algorithms proposed in this thesis are selected. These are CHAID* (Chi-squared Automatic Interaction Detection) and CHAIC. CHAID* is a variation of the CHAID algorithm that, among other things, can handle continuous variables and prunes the decision trees. CHAIC is created by applying the same modifications made to C4.4, to CHAID*. This results in three new consolidated algorithms CTCHAID, CTC44 and CTCHAIC. For clarity, the original CTC algorithm (based on C4.5) is referred to as CTC45 after this point. CHAID* and CHAIC stratify the values of discrete variables to combine several values into a single branch. The consolidation of these two algorithms required figuring out a way of consolidating the individual stratification proposals computed from each sample. This handicap was solved by creating an average contingency table and providing this average table to the algorithm developed by Kass to group the values. The second experiment compares the four base decision tree algorithms and their consolidated versions with the set of 16 genetics-based and five classical algorithms used in the reference work.

Depending on the classification context, the performance of consolidated

algorithms changes. For two out of three context the four consolidated algorithms perform competitively. In one of such contexts, four of the first five ranking positions belong to consolidated algorithms. In another context, almost all consolidated algorithms rank in the top half, above their respective base algorithm. In the third context, base decision trees rank better than their consolidated counterparts but without significant differences if kappa is used as a performance measure context. When comparing all algorithms globally, taking the results of the 96 datasets into account, CTC45 ranks first closely followed by the rest of consolidated algorithms showing no statistically significant differences. All base decision tree algorithms show a more robust behavior than their consolidated counterpart.

6.1.1.3 Conclusions about the Contributions to the Effect of Pruning on Decision Trees and their Consolidated Version

It is not uncommon for the pruning process to fully prune decision trees into a single root node in presence of class imbalance, as some branches of decision trees are very specialized to identify a very small subset of instances. These fully pruned trees do not offer any kind of explanation because all instances are simply classified as majority class.

In this thesis, a new pruning strategy is proposed. This new pruning strategy is called Not Root Tree strategy (NRT), and consists on using the unpruned decision tree if pruning results in a root node tree. The base and consolidated decision trees on the first and second experiments of the thesis follow this strategy. The third experiment compares the performance of decision trees by using different pruning strategies: always pruning, never pruning, NRT, and the same strategy followed by Probability Estimation Trees (PET) such as C4.4 and CHAIC. This comparison is performed independently for C4.5 and CTC45 on one side, and CHAID* and CTCHAID on another.

Results for these experiments show the NRT strategy to be the best strategy in most cases. The results of NRT are usually equal or better than using pruned trees, except for standard classification using kappa as measure. For imbalanced classification with no preprocessing, unpruned and PET trees rank better, at the cost of a higher complexity, as unpruned trees are, by definition, equally complex or more complex than pruned trees. On a global comparison NRT places better for C4.5-based trees while unpruned trees rank better for CHAID*-based trees because unpruned CTCHAID trees distance themselves much more from the nearest competitor on the SMOTE-less two class classification context.

6.1.2 Conclusions for PART-like ruleset algorithms

One of the most widely-used ruleset induction algorithms is PART. This algorithm combines two ruleset induction paradigms: extracting rules from trees and separate-and-conquer strategies. It creates a partial decision tree (hence the name), extracts a rule from the decisions between the root node and the most populous treated leaf, and repeats the process until the ruleset covers the

entire training sample. This thesis proposes multiple algorithms derived from the PART algorithm. These algorithms are referred to PART-like algorithms. The conclusions of this thesis regarding the PART-like ruleset algorithms can be organized in three groups: conclusions about the contributions to modifying the way partial trees are built and used, about the contributions to using fully developed trees instead of partial trees, and about the contributions to using new base algorithms to create PART-like algorithms.

6.1.2.1 Conclusions about the Contributions to Modifying the Way Partial Trees are Built

Four key aspects of PART's development process were identified and an alternative is proposed to how the algorithm originally works on those aspects. PART builds partial decision trees following a Hill Climbing (HC) strategy to determine the order in which nodes are treated and expanded. The modifications proposed in this thesis consist of the following: switching from Hill Climbing to a Best-First (BF) search strategy; considering all leaf-like nodes (AL), and not just treated leaves (TL) to extract the rule in each iteration; not prioritizing the analysis of pure nodes (DP) instead of treating them first (PP); and pruning the partial trees before extracting the rule (PR) instead of keeping them unpruned (NP). This leads to 16 different variants, one of which is the original PART algorithm. Following the study detailed in Appendix H a variant is selected. This variant is the complete opposite to the original PART algorithm in terms of the four identified aspects. It uses a Best-First to select the next node of the partial tree to be developed, considers all leaves as candidates to extract a rule, prunes partial trees, and does not prioritize pure nodes. This variant is named BFPART, and according to the aforementioned study, it creates classifiers with better discriminating capacity and lower structural complexity than PART.

6.1.2.2 Conclusions about the Contributions to using fully developed trees

Based on the notion of developing partial trees further like BFPART does, a second proposal is presented. This proposal completely departs from the idea of using partial trees to create rulesets. Hence the name, UnPART, as the trees are not really partial. This proposal removes the need for three of the four key criteria identified for PART as using fully developed trees removes the need of determining the order in which noes are treated during the tree building process, and all leaf-like nodes are actual true treated leaves once the tree is fully built. The only remaining criterion is whether trees should be pruned or not before extracting a rule. Not pruning a tree results in more complex classifiers as each unpruned rule has at least the same number of decisions as the pruned rule (potentially more), and unpruned rules cover less examples, which in consequence, results in more rules being developed until the ruleset covers the entire training sample. As a result, computational cost increases and could incur in potential overfitting diminishing the discriminating capacity of

the classifiers. In consequence, UnPART prunes trees before extracting a rule.

In this thesis, C4.5-based PART-like algorithms (PART_C45, BFPART_C45, and UnPART_C45) and C4.5 are compared using discriminating capacity (using kappa and GM), structural complexity (using the number of rules and the average number of conditions per rule), and computational cost (using time) metrics to evaluate the algorithms. UnPART_C45 achieves the best discriminating capacity if kappa and GM are used, showing significant improvement compared to PART_C45, and second best (after PART_C45) for AUC. UnPART_C45 also creates the simplest classifiers according to both metrics, creating models significantly simpler than C4.5, and also PART for one of the metrics. For computational cost, C4.5 is significantly faster than any other algorithm. However, C4.5 performs significantly worse for almost every other metric. By combining the results for all metrics, UnPART_C45 is determined to be the best performing algorithm among the compared options.

6.1.2.3 Conclusions about the Contributions to using other decision tree algorithms as base

This thesis also proposes replacing C4.5 as the base algorithm for PART, BFPART and UnPART. The chosen algorithm to replace C4.5 is CHAID*, due to its similarity to the C4.5 algorithm, which eases the integration into PART's construction process. In order to achieve this, a suitable replacement for entropy had to be found to prioritize treating one node over another when constructing partial trees for PART and BFPART. As entropy is a measure closely related to how C4.5 decides the best split for a node, a similar measure was chosen for CHAID*, the p-value of the best possible split for that node. In cases where no valid split can be found the p-value of the parent node's best split is used. UnPART algorithms do not need any p-value as the full tree is developed. In order to distinguish all variants, C4.5-based algorithms receive the *_C45* suffix (PART_C45, BFPART_C45, and UnPART_C45) whereas CHAID*-based algorithms get the *_CHD* suffix (PART_CHD, BFPART_CHD, and UnPART_CHD). Thus, it can be said that this dissertation proposes five PART-like algorithms.

Following the experiment of the previous subsection, CHAID*-based variants are added to the comparison. UnPART_C45 and PART_C45 still achieve the best results for discriminating capacity, significantly better than any CHAID*-based variants. For structural complexity, however, UnPART_CHD achieves the best rank for both metrics, creating significantly simpler classifiers than any C4.5-based algorithm. For execution time, all C4.5-based algorithms are significantly faster than any CHAID*-based algorithm.

One final experiment further explored the contributions of the previous subsection and this one. The discriminating capacity of the eight PART-like algorithms is compared to the 16 genetics-based and five classical rule and decision tree induction algorithms. In this case, UnPART_C45 performs best among the compared algorithms, showing statistically significant differences with six. Both UnPART_C45 and BFPART_C45 (ranking second) perform better than their

base algorithm (C4.5), while all CHAID*-based PART-like algorithms perform better than CHAID*.

Summarizing the three groups of conclusions, UnPART_C45 is an algorithm that should be considered for applications where the understability of the classifiers is crucial, as it does not only create classifiers with the best discriminating capacity compared to a substantial number of ruleset and tree induction algorithms, but it also creates models that are significantly simpler than those built by PART and C4.5. If model simplicity is of value for the problem, UnPART_CHD can also be considered as its classifiers are significantly simpler, at the cost of a lower discriminating capacity and greater execution time.

6.1.3 Conclusions for the consolidation of decision trees and PART-like rulesets

When comparing the 16 genetics-based and five classical algorithms from the reference work to the PART-like ruleset algorithms, and consolidated decision trees with their optimal configuration, results show that the three best ranking positions are occupied by CTC45, CTCHAID, and UnPART_C45, above the rest of the compared algorithms. CTC45 shows statistically significantly better discriminating capacity than 13 of the algorithms, including all CHAID*-based algorithms.

6.2 Further Work

There are several lines of work that remain open in relation to the outcomes presented in this dissertation. This section is further divided into two subsections. Section 6.2.1 outlines further work on consolidation while Section 6.2.2 does the same for PART-like algorithms.

6.2.1 Further Work in the field of Consolidation

Further work in the field of consolidation could focus on the different extracted conclusions. Four research avenues have been identified:

- Using samples of different class distributions.
- Using other resampling techniques.
- Applying consolidation to other types of algorithms.
- Exploring new strategies of when to apply pruning to decision trees.

First, the use of *coverage*-based resampling could be extended to other types of subsamples: bootstrap samples, samples with other shifted class distributions... One interesting approach would be to test different degrees of class imbalance in order to reach an improvement of the GM metric by trying to increase the true-negative rate without sacrificing much of the initial true-positive

rate. This might be achievable by creating samples that are somewhere between stratified and fully balanced samples. Some sort of a controlled imbalance.

Second, research could also focus on the type of resampling techniques used. The methodology used in this thesis uses SMOTE for oversampling the minority class. When samples need to be reduced in size to create balanced samples, the majority class is randomly undersampled. These are two of the most widely used oversampling and undersampling techniques in the literature. However, some other techniques such as SMOTE+ENN (Edited Nearest Neighbors) or genetics-based approaches such as EUSCHC (Evolutionary UnderSampling CHC [51]) could be integrated into the resampling process for consolidated trees.

Third, now that research shows consolidation can be successfully applied to algorithms other than C4.5, it would be interesting to make an even bigger jump and trying to apply the consolidation methodology to other types of classification algorithms. The ideal subset of algorithms to start from would be ruleset algorithms. One of the classical rule induction algorithms used by the reference work and this dissertation, RIPPER, shows great performance in the context of class imbalance. A better performance that C4.5's. As the best performing algorithms of that context are precisely consolidated algorithms, the consolidation process could give RIPPER a chance to top decision tree-based consolidated algorithms. Another way to use consolidation on rule induction algorithms would use the other type of algorithms discussed in this thesis, PART-like algorithms. Instead of using regular decision trees to build PART, BFPART and UnPART trees, consolidated trees could be used to extract rules. One thing to keep in mind with this approach is that both consolidation, and PART-like algorithms multiply the computational cost to build the classifiers.

Finally, this dissertation concludes that NRT is the optimal pruning strategy for base and consolidated decision trees solely based on the results for discriminating capacity. Using unpruned trees has a clear impact on the complexity of the models. It would be interesting to analyze the discriminating capacity / structural complexity trade-off of using different pruning strategies.

An additional avenue, not explored by the research presented in this thesis, are partially consolidating algorithms. The current consolidation technique applies to the full tree-building phase. However, this would not be necessary. Decision trees could be developed using the consolidation technique to a certain point. This would result in smaller consolidated trees with leaves potentially able to expand further. Each of these leaves would contain multiple small samples. One for each of the samples used in the consolidation process. These multiple samples could be used to grow classification models on each leaf. The simplest approach would be to generate a Bagging model on each leaf so that the final model would be part consolidated decision tree, and multiple Bagging classifiers. In fact, the classification model could be unrelated to decision trees, using the samples on each leaf to create an independent classifier, unrelated to the algorithm used to create the partially consolidated tree, in a similar way to how NBTree [104] works by developing a decision tree and then placing a Naive-Bayes classifier on each leaf.

6.2.2 Further Work in the field of PART-like algorithms

Regarding future work in the field of PART-like algorithms, three research avenues have been identified:

- Resampling the training data between rules in a ruleset.
- Exploring ways to determine the stop point of developing partial decision trees sooner.
- Creating ensembles of UnPART rulesets.

First, an interesting proposal would be to use resampling techniques between the iterations of these algorithms. Right now, all of these algorithms just use the instances not covered by the already existing ruleset to create a new tree. However, it could be possible to improve the discriminating capacity of these algorithms by applying resampling techniques to the leftover examples between iterations of the algorithms, like ensemble-based algorithms such as SMOTE-Bagging or RUSBoost do.

Second, results of this thesis clearly indicate that UnPART's biggest weakness is its execution time. This is expected as for the datasets used in this thesis the average number of rules per UnPART ruleset (thus, the number of decision trees to be built) is nine. It could be hypothesized that using the node size as indicator of the next node to be developed, and stopping the process when the biggest node is decided not to be split should yield the same rule, as this biggest node would be used for the rule. Preliminary results show that this variant could potentially be faster than C4.5. Future work could test whether this replacement for UnPART could really maintain the same discriminating capacity, while being faster than C4.5. Also, consolidated trees have shown to stand out when used against imbalanced datasets. As the imbalanced context is where UnPART seems to perform the worst, using consolidated trees as base trees for UnPART could be a viable choice to improve the performance in this context.

Finally, for applications where time and structural complexity are not critical, UnPART rulesets could be used as base for ensemble algorithms (Bagging, Boosting...), as it is very common to use C4.5 as base algorithms for ensembles, and these results show that UnPART has a greater discriminating capacity than C4.5. It is possible that this increase in discriminating capacity might translate to ensemble classifiers as well.

6.3 Related publications

The majority of the work presented in this dissertation has already been published. The following is a list of current publications derived from the thesis.

- International journals:

  - Igor Ibarguren, Jesús M. Pérez, Javier Muguerza, Ibai Gurrutxaga, and Olatz Arbelaitz. Coverage-based resampling: Building robust consolidated decision trees. *Knowledge-Based Systems*, 79, 51-67, 2015.
  - Igor Ibarguren, Jesús M. Pérez, Javier Muguerza, and Ibai Gurrutxaga. CT< *DT* >: Extending the application of the consolidation methodology even further. *Expert Systems*, 34 (5), 2017.
  - Igor Ibarguren, Aritz Lasarguren, Jesús M. Pérez, Javier Muguerza, Ibai Gurrutxaga, and Olatz Arbelaitz. BFPART: Best-First PART. *Information Sciences*, 367, 927-952, 2017.
  - Igor Ibarguren, Jesús M. Pérez, Javier Muguerza, Ibai Gurrutxaga, and Olatz Arbelaitz. UnPART: PART without the 'partial' condition of it, *Information Sciences*, 465, 505-522, 2018.

- National journals:
  - Igor Ibarguren, Jesús M. Pérez, and Javier Muguerza. J48Consolidated WEKA paketea, adibide ezohikoen patroiak identifikatzeko tresna (in Basque). *EKAIA Euskal Herriko Unibertsitateko Zientzia eta Teknologia Aldizkaria*, 29, 155-178, 2016.

- International conferences:
  - Igor Ibarguren, Jesús M. Pérez, and Javier Muguerza. CTCHAID: Extending the Application of the Consolidation Methodology. In *Progress in Artificial Intelligence: 17th Portuguese Conference on Artificial Intelligence, EPIA 2015*, 572-577, 2015.
  - Igor Ibarguren, Jesús M. Pérez, Javier Muguerza, Daniel Rodriguez, and Rachel Harrison. The Consolidated Tree Construction algorithm in imbalanced defect prediction datasets. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2656-2660, 2017.

- National conferences:
  - Igor Ibarguren, Jesús M. Pérez, Javier Muguerza, Ibai Gurrutxaga, and Olatz Arbelaitz. Remuestreo basado en coverage: construyendo árboles de decisión consolidados robustos (key work). In *Actas de la XVI Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA)*, 2015.

- Technical reports:
  - Igor Ibarguren, Jesús M. Pérez, Javier Muguerza, Ibai Gurrutxaga, and Olatz Arbelaitz. An update of the J48Consolidated WEKA's class: CTC algorithm enhanced with the notion of coverage. Technical Report EHU-KAT-IK-02-14, University of the Basque Country (UPV/EHU), June 2014.

Bibliography

[1] H. Abbasian, C. Drummond, N. Japkowicz, and S. Matwin. Inner ensembles: Using ensemble methods inside the learning algorithm. In H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, editors, *Machine Learning and Knowledge Discovery in Databases*, number 8190 in Lecture Notes in Computer Science, pages 33–48. Springer Berlin Heidelberg, Jan. 2013.

[2] J. Abellán and A. R. Masegosa. Bagging schemes on the presence of class noise in classification. *Expert Systems with Applications*, 39(8):6827 – 6837, 2012.

[3] J. S. Aguilar-Ruiz, R. Giraldez, and J. C. Riquelme. Natural encoding for evolutionary supervised learning. *IEEE Transactions on Evolutionary Computation*, 11(4):466–479, Aug 2007.

[4] I. Albisua, O. Arbelaitz, I. Gurrutxaga, A. Lasarguren, J. Muguerza, and J. M. Pérez. The quest for the optimal class distribution: an approach for enhancing the effectiveness of learning via resampling methods for imbalanced data sets. *Progress in Artificial Intelligence*, 2(1):45–63, 2013.

[5] I. Albisua, O. Arbelaitz, I. Gurrutxaga, J. I. Martín, J. Muguerza, J. M. Pérez, and I. Perona. Obtaining optimal class distribution for decision trees: Comparative analysis of CTC and C4.5. In P. Meseguer, L. Mandow, and R. M. Gasca, editors, *Current Topics in Artificial Intelligence*, number 5988 in Lecture Notes in Computer Science, pages 101–110. Springer Berlin Heidelberg, Jan. 2010.

[6] I. Albisua, O. Arbelaitz, I. Gurrutxaga, J. Muguerza, and J. M. Pérez. C4.5 consolidation process: An alternative to intelligent oversampling methods in class imbalance problems. In J. Lozano, J. GÃ¡mez, and J. Moreno, editors, *Advances in Artificial Intelligence*, volume 7023 of *Lecture Notes in Computer Science*, pages 74–83. Springer Berlin Heidelberg, 2011.

[7] I. Albisua, J. Muguerza, and J. M. Pérez. SMOTE versus submuestreo aleatorio en la búsqueda de la distribución de clases óptima. In *Actas del V Simposio de Teoría y Aplicaciones de Minería de Datos (TAMIDA 2010)*, pages 111–120, Valencia, Spain, 2010.

[8] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17(2-3):255–287, 2011.

[9] J. Alcalá-Fdez, L. Sanchez, S. Garcia, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, et al. KEEL: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, 13(3):307–318, 2009.

[10] J. M. Alonso and L. Magdalena. Special issue on interpretable fuzzy systems. *Information Sciences*, 181(20):4331 – 4339, 2011.

[11] S. ANGOSS. *KnowledgeSEEKEER for Windows*. ANGOSS Softwate International Limited, 1995.

[12] O. Arbelaitz, I. Gurrutxaga, J. Infante, J. Muguerza, and J. M. Pérez. CTC: Competitive in an analysis of GMBL algorithms for rule induction in imbalanced datasets. In *Actas de la XV Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 2013)*, pages 19–28, Madrid, Spain, 2013.

[13] O. Arbelaitz, I. Gurrutxaga, A. Lojo, J. Muguerza, J. M. Pérez, and I. Perona. Web usage and content mining to extract knowledge for modelling the users of the bidasoa turismo website and to adapt it. *Expert Systems with Applications*, 40(18):7478 – 7491, 2013.

[14] W.-H. Au, K. C. C. Chan, and X. Yao. A novel evolutionary data mining algorithm with applications to churn prediction. *IEEE Transactions on Evolutionary Computation*, 7(6):532–545, Dec 2003.

[15] J. Bacardit, D. E. Goldberg, and M. V. Butz. Improving the performance of a pittsburgh learning classifier system using a default rule. In T. Kovacs, X. Llorà, K. Takadama, P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Learning Classifier Systems*, pages 291–307, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[16] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations Newsletter*, 6(1):20–29, June 2004.

[17] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1-2):105–139, July 1999.

[18] J. M. Benítez, N. García-Pedrajas, and F. Herrera. Special issue on "new trends in data mining" NTDM. *Knowledge-Based Systems*, 25(1):1–2, Feb. 2012.

[19] B. Bergmann and G. Hommel. Improvements of general multiple test procedures for redundant systems of hypotheses. In P. Bauer, G. Hommel, and E. Sonnemann, editors, *Multiple Hypothesenprüfung / Multiple Hypotheses Testing*, pages 100–115, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[20] E. Bernadó-Mansilla and J. M. Garrell-Guiu. Accuracy-based learning classifier systems: Models, analysis and applications to classification tasks. *Evolutionary Computation*, 11(3):209–238, 2003.

[21] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, August 1996.

[22] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Taylor & Francis, Jan. 1984.

[23] P. L. Bühlmann. Bagging, subagging and bragging for improving some prediction algorithms. In *Research report/Seminar für Statistik, Eidgenössische Technische Hochschule (ETH)*, volume 113. Seminar für Statistik, Eidgenössische Technische Hochschule (ETH), Zürich, 2003.

[24] J. Burez and D. V. d. Poel. Handling class imbalance in customer churn prediction. *Expert Systems with Applications*, 36(3, Part 1):4626 – 4636, 2009.

[25] B. Calvo and G. Santafé Rodrigo. scmamp: statistical comparison of multiple algorithms in multiple problems. *The R Journal, Vol. 8/1, Aug. 2016*, 2016.

[26] E. Cantu-Paz and C. Kamath. Inducing oblique decision trees with evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7(1):54–68, Feb 2003.

[27] D. R. Carvalho and A. A. Freitas. A hybrid decision tree/genetic algorithm method for data mining. *Inf. Sci.*, 163(1-3):13–35, June 2004.

[28] C. Catal. Software fault prediction: A literature review and current trends. *Expert Systems with Applications*, 38(4):4626 – 4636, 2011.

[29] N. Chawla, A. Lazarevic, L. Hall, and K. Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In N. Lavrač, D. Gamberger, L. Todorovski, and H. Blockeel, editors, *Knowledge Discovery in Databases: PKDD 2003*, volume 2838 of *Lecture Notes in Computer Science*, pages 107–119. Springer Berlin Heidelberg, 2003.

[30] N. V. Chawla. C4.5 and imbalanced data sets: investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In *In Proceedings of the ICML'03 Workshop on Class Imbalances*, 2003.

[31] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16(1):321–357, June 2002.

[32] N. V. Chawla, L. O. Hall, K. W. Bowyer, T. E. Moore, and W. P. Kegelmeyer. Distributed pasting of small votes. In *Proceedings of the Third International Workshop on Multiple Classifier Systems*, MCS '02, pages 52–61, London, UK, UK, 2002. Springer-Verlag.

[33] N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explorations Newsletter*, 6(1):1–6, June 2004.

[34] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, Mar 1989.

[35] J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[36] W. W. Cohen. Fast effective rule induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, page 115–123. Morgan Kaufmann, 1995.

[37] A. L. Corcoran and S. Sen. Using real-valued genetic algorithms to evolve rule sets for classification. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pages 120–124 vol.1, Jun 1994.

[38] M. Craven and J. Shavlik. Extracting comprehensible concept representations from trained neural networks. In *Proceedings of IJCAI'95, Workshop on Comprehensibility in Machine Learning*, pages 1–15, 1995.

[39] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[40] J. Derrac, S. García, D. Molina, and F. Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3 – 18, 2011.

[41] B. Desmet and V. Hoste. Online suicide prevention through optimised text classification. *Information Sciences*, 439-440:61 – 78, 2018.

[42] T. G. Dietterich. Machine-learning research: four current directions. *AI Magazine*, 18:97–136, 1997.

[43] P. Domingos. Knowledge acquisition from examples via multiple models. In *In Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*, page 98–106. Morgan Kaufmann, 1997.

[44] P. Domingos. MetaCost: A general method for making classifiers cost-sensitive. In *In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 155–164. ACM Press, 1999.

[45] P. Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, Oct. 2012.

[46] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[47] F. K. Došilović, M. Brčić, and N. Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 0210–0215. IEEE, 2018.

[48] G. Douzas, F. Bacao, and F. Last. Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *Information Sciences*, 465:1–20, 2018.

[49] O. J. Dunn. Multiple comparisons among means. *American Statistical Association*, pages 52–64, 1961.

[50] C. Elkan. The foundations of cost-sensitive learning. In *In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 973–978, 2001.

[51] L. J. Eshelman. The chc adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In G. J. RAWLINS, editor, *Foundations of Genetic Algorithms*, volume 1 of *Foundations of Genetic Algorithms*, pages 265 – 283. Elsevier, 1991.

[52] T. Fawcett. ROC graphs: Notes and practical considerations for researchers. *Machine learning*, 31:1–38, 2004.

[53] A. Fernández, S. García, J. Luengo, E. Bernadó-Mansilla, and F. Herrera. Genetics-based machine learning for rule induction: State of the art,taxonomy and comparative study. *IEEE Transactions on Evolutionary Computation*, 14(6):913 –941, dec. 2010.

[54] R. Fisher. *Statistical methods and scientific inference*. Hafner Press, 1973.

[55] E. Frank and I. Witten. Generating accurate rule sets without global optimization. In *Proc. of the 15th Int. Conference on Machine Learning. Ed. Shavlik,J.* Morgan Kaufmann, 1998.

[56] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *In Proceedings of the Thirteenth International Conference on Machine Learning*, page 148–156, 1996.

[57] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

[58] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[59] M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[60] H. Frigui and R. Krishnapuram. A robust competitive clustering algorithm with applications in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):450–465, May 1999.

[61] J. Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, 1999.

[62] M. Gacto, R. Alcalá, and F. Herrera. Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. *Information Sciences*, 181(20):4340 – 4360, 2011. Special Issue on Interpretable Fuzzy Systems.

[63] S. García, A. Fernández, and F. Herrera. Enhancing the effectiveness and interpretability of decision tree and rule induction classifiers with evolutionary training set selection over imbalanced problems. *Applied Soft Computing*, 9:1304–1314, 2009.

[64] S. García, A. Fernández, J. Luengo, and F. Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044–2064, May 2010.

[65] S. García and F. Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

[66] S. García and F. Herrera. Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evol. Comput.*, 17(3):275–306, Sept. 2009.

[67] V. García, J. S. Sánchez, R. Martín-Félez, and R. A. Mollineda. Surrounding neighborhood-based smote for learning from imbalanced data sets. *Progress in Artificial Intelligence*, 1(4):347–362, 2012.

[68] N. García-Pedrajas, J. Pérez-Rodríguez, M. García-Pedrajas, D. Ortiz-Boyer, and C. Fyfe. Class imbalance methods for translation initiation site recognition in DNA sequences. *Knowledge-Based Systems*, 25(1):22–34, Feb. 2012.

[69] B. Goodman and S. Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI magazine*, 38(3):50–57, 2017.

[70] J. B. Gray and G. Fan. Classification tree analysis using target. *Computational Statistics & Data Analysis*, 52(3):1362 – 1372, 2008.

[71] D. P. Greene and S. F. Smith. Competition-based induction of decision models from examples. *Machine Learning*, 13(2):229–257, Nov 1993.

[72] S.-U. Guan and F. Zhu. An incremental approach to genetic-algorithms-based classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(2):227–239, April 2005.

[73] I. Gurrutxaga, O. Arbelaitz, J. M. Pérez, J. I. Martín, and J. Muguerza. The effect of the used resampling technique and number of samples in consolidated trees' construction algorithm. In *IADIS International Conference Applied Computing*, pages 83–90, 2006.

[74] I. Gurrutxaga, J. M. Pérez, O. Arbelaitz, J. Muguerza, J. I. Martín, and A. Ansuategi. CTC: An alternative to extract explanation from Bagging. In D. Borrajo, L. Castillo, and J. M. Corchado, editors, *Current Topics in Artificial Intelligence*, number 4788 in Lecture Notes in Computer Science, pages 90–99. Springer Berlin Heidelberg, Jan. 2007.

[75] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *SIGKDD Explorations Newsletter*, 11:10–18, November 2009.

[76] H. Han, W.-Y. Wang, and B.-H. Mao. Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In D.-S. Huang, X.-P. Zhang, and G.-B. Huang, editors, *Advances in Intelligent Computing*, number 3644 in Lecture Notes in Computer Science, pages 878–887. Springer Berlin Heidelberg, Jan. 2005.

[77] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(10):993–1001, 1990.

[78] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, corrected edition, July 2003.

[79] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, Sept. 2009.

[80] J. Hernández Orallo, M. J. Ramírez Quintana, and C. Ferri Ramírez. *Introducción a la Minería de Datos*. Pearson Educación, 2004.

[81] T. K. Ho. Random decision forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, pages 278–, Washington, DC, USA, 1995. IEEE Computer Society.

[82] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.

[83] J. Huang and C. X. Ling. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):299–310, 2005.

[84] E. B. Hunt, J. Marin, and P. J. Stone. *Experiments in induction*. Academic Press, 1966.

[85] I. Ibarguren, A. Lasarguren, J. M. Pérez, J. Muguerza, I. Gurrutxaga, and O. Arbelaitz. BFPART: Best-First PART. *Information Sciences*, 367–368:927 – 952, 2016.

[86] I. Ibarguren, J. M. Pérez, J. Muguerza, and I. Gurrutxaga. CT<DT>: Extending the application of the consolidation methodology even further. *Expert Systems*, pages e12212–n/a, 2017.

[87] I. Ibarguren, J. M. Pérez, J. Muguerza, I. Gurrutxaga, and O. Arbelaitz. Coverage-based resampling: Building robust consolidated decision trees. *Knowledge-Based Systems*, 79:51–67, 2015.

[88] R. L. Iman and J. M. Davenport. Approximations of the critical region of the fbietkan statistic. *Communications in Statistics - Theory and Methods*, 9(6):571–595, 1980.

[89] C. Z. Janikow. A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning*, 13(2):189–228, Nov 1993.

[90] N. Japkowicz et al. Learning from imbalanced data sets: a comparison of various strategies. In *AAAI workshop on learning from imbalanced data sets*, volume 68, pages 10–15. Menlo Park, CA, 2000.

[91] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intell. Data Anal.*, 6(5):429–449, Oct. 2002.

[92] L. A. Jeni, J. F. Cohn, and F. De La Torre. Facing imbalanced data–recommendations for the use of performance metrics. In *Proceedings of the 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, ACII '13, pages 245–251, Washington, DC, USA, 2013. IEEE Computer Society.

[93] L. Jiao, J. Liu, and W. Zhong. An organizational coevolutionary algorithm for classification. *IEEE Transactions on Evolutionary Computation*, 10(1):67–80, Feb 2006.

[94] T. Jo and N. Japkowicz. Class imbalances versus small disjuncts. *SIGKDD Explor. Newsl.*, 6(1):40–49, June 2004.

[95] U. Johansson, L. Niklasson, and R. Köning. Accuracy vs. comprehensibility in data mining models. In *Proceedings of the Seventh International Conference on Information Fusion*, pages 295–300, 2004.

[96] M. Joshi, V. Kumar, and R. C. Agarwal. Evaluating boosting algorithms to classify rare classes: comparison and improvements. In *ICDM 2001, Proceedings IEEE International Conference on Data Mining, 2001*, pages 257–264, 2001.

[97] G. V. Kass. Significance testing in automatic interaction detection (a.i.d.). *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 24 (2):178–189, 1975.

[98] G. V. Kass. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29(2):119, 1980.

[99] M. Kearns. Thoughts on hypothesis boosting. *Unpublished manuscript*, 1988.

[100] H. Lee, J. Kim, and S. Kim. Gaussian-based SMOTE algorithm for solving skewed class distributions. *International Journal of Fuzzy Logic and Intelligent Systems*, 17(4):229–234, 2017.

[101] J. Luengo, A. Fernández, S. García, and F. Herrera. Addressing data complexity for imbalanced data sets: analysis of smote-based oversampling and evolutionary undersampling. *Soft Computing*, 15(10):1909–1936, Oct 2011.

[102] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774, 2017.

[103] J. Magidson and S. Inc. *SPSS for Windows: CHAID,Release 6.0*. SPSS Incorporated, 1993.

[104] D. Y. Mahmood and M. A. Hussein. Analyzing nb, dt and nbtree intrusion detection algorithms. *Journal of Zankoy Sulaimani-Part A (JZS-A)*, 16(1):1, 2014.

[105] L. Manevitz and M. Yousef. One-class document classification via neural networks. *Neurocomputing*, 70(7–9):1466 – 1481, 2007. Advances in Computational Intelligence and Learning, 14th European Symposium on Artificial Neural Networks 2006.

[106] M. A. Mazurowski, P. A. Habas, J. M. Zurada, J. Y. Lo, J. A. Baker, and G. D. Tourassi. Training neural network classifiers for medical decision making: the effects of imbalanced datasets on classification performance. *Neural networks: the official journal of the International Neural Network Society*, 21(2-3):427–436, Apr. 2008.

[107] R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The multi-purpose incremental learning system aq15 and its testing application to three medical domains. In *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence*, AAAI'86, pages 1041–1045. AAAI Press, 1986.

[108] J. Morgan and J. Sonquist. Problems in the analysis of survey data,and a proposal. *J. Amer. Statistics Ass.*, 58:415–434, 1963.

[109] P. Nemenyi. Distribution-free multiple comparisons. In *Biometrics*, number 2(18), page 263. International Biometric Soc 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005-2210, 1962.

[110] X. Niuniu and L. Yuxun. Review of decision trees. In *Computer Science and Information Technology (ICCSIT),2010 3rd IEEE International Conference on*, volume 5, pages 105 –109, july 2010.

[111] A. Orriols-Puig and E. Bernadó-Mansilla. Evolutionary rule-based systems for imbalanced data sets. *Soft Computing*, 13(3):213–225, 2009.

[112] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Longman Publishing Co.,Inc., Boston,MA,USA, 1984.

[113] K. Pearson. X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175, 1900.

[114] J. M. Pérez. *Árboles consolidados: construcción de un árbol de clasificación basado en múltiples submuestras sin renunciar a la explicación.* PhD thesis, University of the Basque Country, 2006.

[115] J. M. Pérez, I. Albisua, O. Arbelaitz, I. Gurrutxaga, J. Martín, J. Muguerza, and I. Perona. Consolidated trees versus bagging when explanation is required. *Computing*, 89:113–145, 2010.

[116] J. M. Pérez, J. Muguerza, O. Arbelaitz, and I. Gurrutxaga. Consolidated tree construction algorithm: Structurally steady trees. In *Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS-2004)*, pages 14–21, Porto, Portugal, 2004.

[117] J. M. Pérez, J. Muguerza, O. Arbelaitz, and I. Gurrutxaga. A new algorithm to build consolidated trees: study of the error rate and steadiness. In M. Kłopotek, S. Wierzchoń, and K. Trojanowski, editors, *Intelligent Information Processing and Web Mining*, volume 25 of *Advances in Soft Computing*, pages 79–88. Springer Berlin Heidelberg, 2004.

[118] J. M. Pérez, J. Muguerza, O. Arbelaitz, I. Gurrutxaga, and J. I. Martín. Analysis of structural convergence of consolidated trees when resampling is required. In *Proceedings of the 3rd Australasian Data Mining Conference (AusDM04)*, pages 9–21, Cairns, Australia, 2004.

[119] J. M. Pérez, J. Muguerza, O. Arbelaitz, I. Gurrutxaga, and J. I. Martín. Consolidated tree classifier learning in a car insurance fraud detection domain with class imbalance. In S. Singh, M. Singh, C. Apte, and P. Perner, editors, *Pattern Recognition and Data Mining*, number 3686 in Lecture Notes in Computer Science, pages 381–389. Springer Berlin Heidelberg, Jan. 2005.

[120] J. M. Pérez, J. Muguerza, O. Arbelaitz, I. Gurrutxaga, and J. I. Martín. Consolidated trees: Classifiers with stable explanation. a model to achieve the desired stability in explanation. In S. Singh, M. Singh, C. Apte, and P. Perner, editors, *Pattern Recognition and Data Mining*, number 3686 in Lecture Notes in Computer Science, pages 99–107. Springer Berlin Heidelberg, Jan. 2005.

[121] J. M. Pérez, J. Muguerza, O. Arbelaitz, I. Gurrutxaga, and J. I. Martín. Consolidated trees: An analysis of structural convergence. In G. J. Williams and S. J. Simoff, editors, *Data Mining*, number 3755 in Lecture Notes in Computer Science, pages 39–52. Springer Berlin Heidelberg, Jan. 2006.

[122] J. M. Pérez, J. Muguerza, O. Arbelaitz, I. Gurrutxaga, and J. I. Martín. Combining multiple class distribution modified subsamples in a single tree. *Pattern Recognition Letters*, 28(4):414–422, Mar. 2007.

[123] G. Pérez-Caballero, J. Andrade, P. Olmos, Y. Molina, I. Jiménez, J. Durán, C. Fernandez-Lozano, and F. Miguel-Cruz. Authentication of tequilas using pattern recognition and supervised classification. *TrAC Trends in Analytical Chemistry*, 94:117 – 129, 2017.

[124] I. Perona. *Behaviour modelling with data obtained from the Internet and contributions to cluster validation.* PhD thesis, University of the Basque Country, 2016.

[125] F. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(3):199–215, 2003.

[126] P. Pu and L. Chen. Trust building with explanation interfaces. In *Proceedings of the 11th international conference on Intelligent user interfaces*, pages 93–100, 2006.

[127] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, March 1986.

[128] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[129] E. Ramentol, Y. Caballero, R. Bello, and F. Herrera. Smote-rsb *: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using smote and rough sets theory. *Knowledge and Information Systems*, 33(2):245–265, 2012.

[130] D. K. Renuka, T. Hamsapriya, M. R. Chakkaravarthi, and P. L. Surya. Spam classification based on supervised learning using machine learning techniques. In *2011 International Conference on Process Automation, Control and Computing*, pages 1–7, July 2011.

[131] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465 – 471, 1978.

[132] S. J. Russell, P. Norvig, J. F. Candy, J. M. Malik, and D. D. Edwards. *Artificial intelligence: a modern approach*. Prentice-Hall,Inc., Upper Saddle River,NJ,USA, 1996.

[133] J. A. Sáez, J. Luengo, J. Stefanowski, and F. Herrera. Smote–ipf: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. *Information Sciences*, 291:184 – 203, 2015.

[134] G. Schaefer and T. Nakashima. Michigan vs. Pittsburgh Style GA Optimisation of Fuzzy Rule Bases for Gene Expression Analysis. *International Journal of Fuzzy System Applications*, 3:60–72, 10 2015.

[135] R. E. Schapire. The strength of weak learnability. *Mach. Learn.*, 5(2):197–227, July 1990.

[136] C. Seiffert, T. Khoshgoftaar, J. Van Hulse, and A. Napolitano. RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 40(1):185–197, 2010.

[137] C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, and A. Folleco. An empirical study of the classification performance of learners on imbalanced and noisy software quality data. *Information Sciences*, 259(0):571 – 595, 2014.

[138] J. P. Shaffer. Modified sequentially rejective multiple test procedures. *Journal of the American Statistical Association*, 81(395):826–831, 1986.

[139] A. Shepitsen, J. Gemmell, B. Mobasher, and R. Burke. Personalized recommendation in social tagging systems using hierarchical clustering. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 259–266, New York, NY, USA, 2008. ACM.

[140] D. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures: Third Edition*. CRC Press, 2003.

[141] H. Shi. Best-first decision tree learning. Master's thesis, University of Waikato, Hamilton,NZ, 2007. COMP594.

[142] B. Sierra. *Aprendizaje automático: conceptos básicos y avanzados : aspectos prácticos utilizando el software Weka*. Pearson Prentice Hall, 2006.

[143] K. C. Tan, Q. Yu, and J. H. Ang. A coevolutionary algorithm for rules discovery in data mining. *International Journal of Systems Science*, 37(12):835–864, 2006.

[144] N. Tomašev and D. Mladenić. Class imbalance and the curse of minority hubs. *Knowledge-Based Systems*, 53:157–172, Nov. 2013.

[145] S. Tsang, B. Kao, K. Yip, W.-S. Ho, and S. D. Lee. Decision trees for uncertain data. *IEEE Transactions on Knowledge and Data Engineering*, 23(1):64 –78, 2011.

[146] P. Turney. Technical note: Bias and the quantification of stability. *Machine Learning*, 20(1-2):23–33, July 1995.

[147] G. Venturini. Sia: A supervised inductive algorithm with genetic search for learning attributes based concepts. In P. B. Brazdil, editor, *Machine Learning: ECML-93*, pages 280–296, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[148] S. Wang and X. Yao. Diversity analysis on imbalanced data sets by using ensemble models. In *2009 IEEE Symposium on Computational Intelligence and Data Mining*, pages 324–331, March 2009.

[149] G. I. Webb. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, 40(2):159–196, Aug 2000.

[150] G. M. Weiss and F. Provost. Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–354, Oct. 2003.

[151] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[152] S. W. Wilson. Classifier fitness based on accuracy. *Evol. Comput.*, 3(2):149–175, June 1995.

[153] G. Wu and E. Y. Chang. Class-boundary alignment for imbalanced dataset learning. In *ICML 2003 Workshop on Learning from Imbalanced Data Sets*, pages 49–56, 2003.

[154] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008. 10.1007/s10115-007-0114-2.

[155] S. Xia, Z. Xiong, Y. Luo, L. Dong, and C. Xing. Relative density based support vector machine. *Neurocomputing*, 149:1424 – 1432, 2015.

[156] Q. Yang and X. Wu. 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, 5(4):597–604, 2006.

[157] B. Zadrozny and C. Elkan. Learning and making decisions when costs and probabilities are both unknown. In *In Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, pages 204–213. ACM Press, 2001.

[158] J. Zar. *Biostatistical Analysis*. Prentice Hall, 2010.

[159] G. Zhang. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 30(4):451–462, 2000.

[160] C. Zheng, S. Chen, W. Wang, and J. Lu. Using principal component analysis to solve a class imbalance problem in traffic incident detection. *Mathematical Problems in Engineering*, 2013, 2013.

[161] F. Zhu and S.-U. Guan. Ordered incremental training with genetic algorithms. *International Journal of Intelligent Systems*, 19(12):1239–1256, 2004.

[162] J. Zhu, A. Liapis, S. Risi, R. Bidarra, and G. M. Youngblood. Explainable ai for designers: A human-centered perspective on mixed-initiative co-creation. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2018.

[163] L. Zhuang and H. Dai. Parameter optimization of kernel-based one-class classifier on imbalance text learning. In Q. Yang and G. Webb, editors, *PRICAI 2006: Trends in Artificial Intelligence*, volume 4099 of *Lecture Notes in Computer Science*, pages 434–443. Springer Berlin Heidelberg, 2006.

Part V

Appendices

Appendix A

Summary of Appendices

These appendices mostly include the full result tables for the experiments carried out through the thesis. But before beginning with the experiment results:

- Appendix B details the dataset-by-dataset results of the 16 genetics-based and 6 classical rule induction algorithms from the reference work described in Section 2.10. These results are used as benchmark to test the performance of the contributions presented throughout the thesis.

Part III, Contributions, looks at results from a macro level, either from a point of view from classification contexts (30 - 33 datasets at the same time), or from a global point of view (3 contexts, 96 datasets). The following appendices give micro level detail of the dataset-by-dataset performance of the classifiers. Appendices C through G extend the tables from the contributions to the consolidation of decision tree algorithms:

- Appendix C specifies the N_S values for each dataset and *coverage* value in Experiment 1 of Chapter 3.
- Appendix D displays dataset-by-dataset result tables for the 96 datasets, 2 subsample types and 11 *coverage* values for Experiment 1 of Chapter 3.
- Appendix E extends the analysis of the effect of subsample type and *coverage* value on Imbalanced datasets.
- Appendix F extends the results from Experiment 2 of Chapter 3.
- Appendix G details the results for different pruning strategies on base and consolidated decision trees. This was Experiment 3 of Chapter 3.

Appendices H and I extend the tables from the contributions to PART-like ruleset algorithms:

- Appendix H details a study carried out during the thesis that was left out of the main sections of the dissertation. This work was used to propose four changes to the PART ruleset induction algorithm. This study analyzed 16 variants of the algorithm, and selected the best one, later used on Chapter 4. This work followed an experimental methodology different from the rest of the thesis.
- Appendix I extends the results analyzed on Chapter 4.

Appendix B

Results from the reference work for genetics-based and classical algorithms

This appendix contains the result tables from the reference work used in the comparisons throughout the thesis, and described in Section 2.10.

In tables regarding genetics-based algorithms, columns in bold indicate that algorithms is best for that particular subcategory and, thus, used on context-by-context comparisons. Underlined column headers indicate that algorithm is not the best on that subcategory for that particular context, but it is for one of the other two contexts and, thus, used on global comparisons that combine the results from the three contexts.

In tables regarding the classical algorithms, only the algorithms from the referenced work have been included and not the algorithms proposed in this thesis. In fact, the results published by the reference work for the C4.5 have been omitted as this thesis uses another implementation of the C4.5 algorithm.

	Michigan approach		IRL approach		GCCL Approach			Pittsburgh approach					HEDT approach		
dataset	**XCS**	UCS	**SIA**	HIDER	CORE	**OCEC**	COGIN	GIL	Pitts-GIRLA	DMEL	**GAssist**	OIGA	ILGA	DT-GA	**Oblique-DT**	Target
nursery	**.007**	.0068	**.0085**	.0074	.0006	**.0074**	.0074	.0074	.0032	.0029	**.0086**	.0075	.0059	.0081	**.009**	.0058
abalone	**.0007**	.0008	**.001**	.001	.0003	**.0007**	.001	.0005	.0007	.0002	**.0013**	.0012	.0009	.0006	**.0007**	.0007
ecoli	**.007**	.0069	**.0057**	.0067	.0005	**.0048**	.0042	.0047	.0056	.0017	**.0067**	.0056	.0045	.0067	**.0067**	.0048
lymphography	**.006**	.0054	**.0061**	.0054	.0014	**.0055**	.0054	.0056	.0024	.0005	**.0059**	.0059	.0055	.0036	**.0046**	.0052
car	**.0043**	.0081	**.0086**	0	.0008	**.0055**	.0018	.0064	.0047	.0003	**.0082**	.0047	.0002	.0067	**.0096**	.0056
zoo	**.0085**	.0087	**.0094**	.0094	.001	**.0092**	.0088	.0092	.0011	.0047	**.0092**	.009	.0084	.0092	**.0095**	.0062
flare	**.0039**	.0064	**.0035**	.0067	.0003	**.0063**	.0059	.0053	.0026	.0035	**.0066**	.0064	.0053	.0067	**.0066**	.005
glass	**.0054**	.0051	**.0064**	.0049	.0004	**.0034**	.0037	.0041	.0041	.0006	**.0045**	.0037	.0031	.0057	**.0055**	.0023
cleveland	**.0028**	.0023	**.0017**	.002	.0009	**.0031**	.0021	.0019	.0009	.0011	**.0024**	.0011	.0007	.0022	**.0022**	.0013
dermatology	**.0094**	.0078	**.0079**	.0084	.001	**.0075**	.0082	.008	.0037	.0032	**.009**	.0082	.0082	.0081	**.0092**	.0056
balance	**.0069**	.006	**.0068**	.0051	.0004	**.0051**	.0054	.0034	.0043	.0046	**.006**	.0017	.0019	.0055	**.0083**	.0055
penbased	**.0087**	.009	**.0095**	.0065	.0008	**.0064**	.007	.0043	.0026	.001	**.0062**	.0044	.0042	.0085	**.009**	.0032
newthyroid	**.0088**	.0088	**.0085**	.0083	.0004	**.0071**	.0063	.0069	.0079	.0025	**.0085**	.0084	.0082	.0078	**.0089**	.0072
hepatitis	**.0054**	.0023	**.0013**	.0029	.0021	**.0036**	.0011	.003	.0012	0	**.0042**	.003	.0024	0	**.0042**	.001
contraceptive	**.0028**	.002	**.0019**	.0024	.0004	**.0019**	.0014	.0019	.0027	.0009	**.0028**	.0013	.0006	.0022	**.0019**	.0011
vehicle	**.0064**	.0062	**.005**	.0051	.0005	**.0036**	.0038	.0028	.0027	.0011	**.0055**	.0048	.0045	.0058	**.006**	.0033
haberman	**.0008**	.0017	**.0013**	.0012	.0008	**.0009**	.0003	.0009	.0004	.0008	**.0009**	.0001	.0001	.0015	**.0007**	.0007
wine	**.0095**	.0086	**.0093**	.0065	.0007	**.0059**	.005	.0059	.0065	.0012	**.0089**	.0087	.0083	.0092	**.0088**	.0073
breast	**.0028**	.0029	**.0022**	.0027	.0005	**.0021**	.0016	.002	.0019	.0032	**.0031**	.0029	.0024	.0019	**.0011**	.003
german	**.0031**	.0026	**.0011**	.0019	.0007	**.0024**	.0015	.0031	.0014	.001	**.003**	.0018	.0021	.0028	**.0023**	0
iris	**.0092**	.0085	**.0092**	.0091	.0003	**.0082**	.0074	.0085	.008	.0025	**.0094**	.0091	.009	.009	**.0089**	.0089
wisconsin	**.0093**	.0091	**.0093**	.0092	.0003	**.0091**	.009	.009	.0048	.0037	**.009**	.0082	.0081	.0088	**.0085**	.0091
tictactoe	**.0065**	.008	**.0099**	.0034	.0004	**.0059**	.0082	.0041	.0033	.0025	**.0089**	.0043	.005	.0059	**.0078**	.003
pima	**.0042**	.0041	**.0036**	.0036	.0006	**.0023**	.0012	.0024	0	.0006	**.0039**	.004	.0035	.0042	**.0035**	.0041
magic	**.0056**	.0053	**.0045**	.0038	.0007	**.0016**	.001	.0015	.0051	0	**.0054**	.0048	.0044	.0053	**.0044**	.0045
bupa	**.0027**	.0027	**.0026**	.0023	.0005	.	-.0003	-.0002	.0018	.0001	**.0027**	.0007	.0002	.0027	**.0024**	.0028
heart	**.0056**	.0052	**.0034**	.0044	.0003	**.0056**	.0056	.0054	0	.0055	**.0059**	.0059	.0059	.0053	**.0047**	.0048
australian	**.0071**	.0066	**.0028**	.0064	.0009	**.0072**	.0069	.0069	.0046	.0005	**.007**	.0069	.007	.0068	**.0063**	.0071
crx	**.0071**	.0067	**.0029**	.0062	.001	**.0073**	.0068	.007	.0072	.0005	**.0072**	.007	.0072	.0072	**.006**	.0073
ring	**.0084**	.0059	**.0037**	-.0033	.0016	**.0057**	.0039	.0048	.0022	0	**.0076**	.0075	.0074	.0069	**.006**	.0035
Mean	**.0059**	.0057	**.0052**	.0047	.0007	**.0048**	.0044	.0046	.0033	.0017	**.006**	.005	.0045	.0055	**.0058**	.0043

Table B.1: Results for genetics-based algorithms on standard datasets using the kappa measure

dataset	CART	AQ	CN2	C4.5rules	RIPPER
nursery	.7764	.6853	.6903	.6903	.8386
abalone	.0918	.059	.159	.159	.0935
ecoli	.6367	.5349	.5816	.5816	.6559
lymphography	.5611	.504	.6325	.6325	.5627
car	.5534	.6689	.323	.323	.7591
zoo	.8815	.8855	.8197	.8197	.8828
flare	.6702	.5552	.5596	.5596	.5891
glass	.5738	.4477	.436	.436	.5288
cleveland	.1655	.1288	.0918	.0918	.2068
dermatology	.6567	.8341	.8448	.8448	.8513
balance	.3024	.216	.6069	.6069	.3178
penbased	.5768	.5756	.6318	.6318	.8412
newthyroid	.8807	.6981	.7877	.7877	.8769
hepatitis	.3124	.3916	.2169	.2169	.3191
contraceptive	.2517	.0105	.0652	.0652	.2723
vehicle	.54	.4252	.4083	.4083	.6104
haberman	.0387	.0018	.0051	.0051	.1432
wine	.88	.6885	.7847	.7847	.8504
breast	0	.2166	.1417	.1417	.1716
german	.2591	.1858	.1254	.1254	.251
iris	.93	.722	.86	.86	.896
wisconsin	.8509	.6903	.8984	.8984	.9122
tictactoe	.4222	.8613	.1881	.1881	.9375
pima	.4002	.0862	.1239	.1239	.3795
magic	.448	.1092	.1524	.1524	.499
bupa	.3465	.0184	.0538	.0538	.2839
heart	.4459	.5577	.5097	.5097	.5017
australian	.6947	.3876	.5884	.5884	.63
crx	.6864	.5832	.5751	.5751	.6387
ring	.6675	-.0076	.5907	.5907	.6286
Mean	.5167	.424	.4484	.4484	.5643

Table B.2: Results for classical algorithms on standard datasets using the kappa measure

	Michigan approach		IRL approach	GCCL Approach			Pittsburgh approach					HEDT approach	
dataset	**XCS**	UCS	**SIA**	HIDER	CORE	**OCEC**	COGIN	GIL	Pitts-GIRLA	DMEL	**GAssist**	OIGA	ILGA	DT-GA	**Oblique-DT**	Target
nursery	**.7969**	.7854	**.8948**	.8225	.7159	**.8162**	.8188	.8187	.5356	.3845	**.9023**	.8321	.7237	.8741	**.929**	.7202
abalone	**.1867**	.178	**.1952**	.192	.1929	**.1412**	.2039	.1048	.1852	.0292	**.2445**	.2365	.2072	.1382	**.1656**	.2015
ecoli	**.7899**	.7803	**.6948**	.7631	.634	**.6126**	.6133	.587	.6917	.4388	**.7667**	.6846	.6209	.7673	**.7619**	.6549
lymphography	**.7946**	.7679	**.797**	.766	.6714	**.7633**	.7467	.7658	.6349	.2697	**.7873**	.7684	.7581	.6582	**.7166**	.7517
car	**.7955**	.916	**.9337**	.7002	.7924	**.7752**	.7212	.8058	.7902	.0859	**.9206**	.7978	.7049	.8625	**.9838**	.7771
zoo	**.8888**	.9069	**.9525**	.9543	.9254	**.9366**	.9108	.9391	.3386	.6301	**.9366**	.9263	.8791	.937	**.9605**	.7209
flare	**.5317**	.7212	**.5201**	.7442	.6557	**.7054**	.6788	.6152	.4696	.5097	**.7399**	.7194	.6375	.7472	**.7317**	.6154
glass	**.6647**	.6412	**.7343**	.6391	.5103	**.4972**	.5246	.5499	.5945	.1534	**.6142**	.5494	.5102	.6835	**.6712**	.4411
cleveland	**.5441**	.5325	**.4956**	.5124	.5415	**.5697**	.4976	.435	.5442	.2707	**.5421**	.5152	.5145	.4961	**.4915**	.5299
dermatology	**.9559**	.8441	**.8324**	.8759	.3172	**.8045**	.8565	.8435	.4979	.499	**.9229**	.8598	.8527	.8485	**.9385**	.6615
balance	**.8269**	.7693	**.824**	.7354	.6646	**.7005**	.7491	.5232	.6912	.7091	**.7856**	.5344	.536	.752	**.9056**	.7562
penbased	**.8858**	.9104	**.9573**	.6829	.3338	**.6775**	.7284	.4879	.3302	.1915	**.6584**	.4913	.4718	.8633	**.9115**	.3893
newthyroid	**.946**	.9488	**.9312**	.9209	.9135	**.8614**	.8177	.8419	.9042	.4056	**.9274**	.9219	.9144	.9042	**.9479**	.8679
hepatitis	**.9**	.82	**.84**	.8425	.8	**.79**	.44	.815	.73	.1625	**.8575**	.795	.7675	.815	**.8325**	.7725
contraceptive	**.5406**	.4828	**.4903**	.5231	.4352	**.4517**	.4456	.4345	.5419	.3442	**.5446**	.4725	.4409	.5154	**.4771**	.4519
vehicle	**.727**	.7163	**.6239**	.63	.3955	**.5233**	.511	.4598	.452	.33	**.6646**	.5989	.5738	.6868	**.7033**	.4981
haberman	**.719**	.7209	**.6593**	.7288	.7085	**.6745**	.7314	.6732	.6338	.7254	**.6868**	.717	.7242	.7222	**.6248**	.715
wine	**.9683**	.9054	**.9526**	.7749	.9029	**.727**	.6429	.7291	.7884	.3787	**.9266**	.9101	.8849	.949	**.9208**	.8224
breast	**.7489**	.7323	**.6946**	.7545	.7301	**.654**	.5421	.6268	.7229	.7215	**.7394**	.7359	.7394	.7336	**.6284**	.7134
german	**.7312**	.7186	**.677**	.7104	.7024	**.67**	.4726	.6654	.7126	.6268	**.726**	.711	.7132	.7176	**.6706**	.7
iris	**.944**	.9	**.944**	.9413	.9347	**.8813**	.828	.9	.864	.5027	**.9587**	.9373	.9293	.9333	**.928**	.9293
wisconsin	**.966**	.9614	**.9696**	.964	.9394	**.9581**	.9531	.9555	.7472	.759	**.9555**	.9127	.9095	.9456	**.9306**	.9575
tictactoe	**.8532**	.9209	**.9977**	.6993	.7019	**.8121**	.9123	.6663	.7382	.6094	**.9511**	.749	.7806	.8265	**.9008**	.695
pima	**.7466**	.7393	**.719**	.7346	.7211	**.6963**	.6703	.6979	.5913	.664	**.7327**	.7362	.7211	.7396	**.7083**	.7302
magic	**.8124**	.7931	**.7537**	.7344	.7336	**.6877**	.6723	.6841	.7894	.6491	**.7994**	.7702	.7583	.8011	**.744**	.7576
bupa	**.6603**	.658	**.6383**	.6423	.6023	**.4522**	.5536	.4586	.5942	.4643	**.6481**	.5687	.549	.6464	**.6307**	.6597
heart	**.7852**	.7689	**.6756**	.7237	.7467	**.78**	.7733	.7704	.5333	.7785	**.8007**	.7896	.7881	.7726	**.74**	.7489
australian	**.8559**	.8351	**.653**	.822	.8159	**.8591**	.8455	.8449	.7501	.5768	**.8539**	.8426	.847	.8423	**.8203**	.8554
crx	**.8579**	.835	**.6561**	.8098	.8282	**.8637**	.8359	.8496	.8607	.5678	**.8585**	.846	.857	.86	**.801**	.8637
ring	**.9184**	.7951	**.6862**	.3395	.6114	**.7846**	.6889	.7416	.5989	.5041	**.8814**	.8681	.8605	.8446	**.7989**	.6759
Mean	**.7781**	.7668	**.7465**	.7228	.6726	**.7042**	.6795	.6763	.6286	.4647	**.7778**	.7266	.7058	.7628	**.7658**	.6878

Table B.3: Results for genetics-based algorithms on standard datasets using the accuracy measure

dataset	CART	AQ	CN2	C4.5rules	RIPPER
nursery	.848	.7667	.7917	.8367	.8895
abalone	.2129	.1517	.268	.2097	.1737
ecoli	.7382	.6858	.7145	.777	.7437
lymphography	.7761	.7208	.8108	.7265	.7634
car	.8085	.8682	.7726	.8681	.8828
zoo	.9105	.9145	.871	.931	.9111
flare	.7448	.6593	.666	.7015	.6713
glass	.687	.6059	.6079	.6276	.648
cleveland	.4917	.5126	.5388	.5387	.4404
dermatology	.7322	.8704	.8798	.9079	.8811
balance	.6224	.5712	.7872	.8064	.5069
penbased	.62	.6185	.6691	.872	.8571
newthyroid	.9442	.8428	.907	.9023	.9433
hepatitis	.825	.8725	.8625	.8125	.7975
contraceptive	.5133	.4258	.4447	.5166	.521
vehicle	.6536	.5672	.5579	.6447	.7076
haberman	.6961	.2791	.7026	.696	.5071
wine	.9211	.7947	.8595	.949	.9006
breast	.7076	.7285	.7221	.7071	.6144
german	.701	.7144	.725	.7116	.6666
iris	.9533	.8147	.9067	.94	.9307
wisconsin	.9327	.8333	.9547	.9503	.9596
tictactoe	.7265	.94	.7056	.8601	.9716
pima	.7187	.6703	.6732	.7284	.7005
magic	.7476	.6751	.6866	.7843	.7689
bupa	.6899	.4458	.5942	.6417	.6359
heart	.7259	.7859	.7667	.7778	.7474
australian	.8464	.713	.8014	.8481	.8133
crx	.8439	.7889	.7949	.8539	.8177
ring	.8338	.4922	.7959	.8492	.8141
Mean	.7391	.6777	.728	.7659	.7396

Table B.4: Results for classical algorithms on standard datasets using the accuracy measure

dataset	Michigan approach		IRL approach		GCCL Approach			Pittsburgh approach					HEDT approach		
	XCS	**UCS**	**SIA**	HIDER	CORE	**OCEC**	COGIN	GIL	Pitts-GIRLA	DMEL	**GAssist**	OIGA	ILGA	DT-GA	**Oblique-DT**	Target
abalone19	0	**0**	**0**	0	0	**.6172**	0	.6896	0	.0804	**0**	0	0	0	**.0753**	0
yeast6	0	**.7553**	**.5147**	0	0	**.5424**	.182	.5643	0	0	**0**	0	0	.566	**.628**	0
yeast5	0	**.7814**	**.7771**	0	.0943	**.7804**	.2656	.7944	.2801	0	**.7267**	.4132	.1369	.8367	**.8082**	0
yeast4	0	**.5996**	**.2621**	.063	.0631	**.5831**	.0891	.5891	.1121	0	**.1716**	.1234	.1231	.3992	**.5556**	0
yeast2_vs_8	.6965	**.5657**	**.1**	.6958	.7266	**.7145**	.6546	.7123	.1	.2555	**.6274**	0	0	.1	**.6905**	.7283
glass5		**.3397**	**.6779**	.3287	.7347	**.8834**	.1975	.7201	0	0	**.1397**	.6762	0	.4225	**.9365**	0
abalone9_vs_18	.1414	**.4136**	**.4017**	.1855	.3273	**.4865**	.2056	.535	.2119	.4467	**.4041**	.2036	.0707	.4478	**.5411**	0
glass4	.8597	**.3612**	**.9541**	.4155	.5647	**.6972**	.3579	.6651	.114	.0763	**.6251**	.2767	.4849	.3009	**.627**	0
ecoli4	.4437	**.6873**	**.6521**	.695	.784	**.7606**	.2718	.7338	.6374	.0756	**.7802**	.7528	.58	.8233	**.8809**	.677
glass2		**.0961**	**.1934**	0	.1	**.6599**	0	.6711	0	0	**.**	0	0	.5259	**.3205**	0
vowel0	.9081	**.9668**	**1**	.7637	.726	**.6238**	.3967	.6277	.6033	.1686	**.936**	.8452	.8171	.9683	**.9559**	.7107
page-blocks0	.8458	**.8504**	**.8659**	.8927	.6492	**.7514**	.5501	.7473	.4575	0	**.8087**	.8162	.6249	.9218	**.9007**	.5999
ecoli3	.3191	**.4629**	**.6169**	.4504	.4879	**.6329**	.567	.6632	.4214	.073	**.6711**	.5084	.4023	.7268	**.6715**	0
yeast3	.1718	**.8946**	**.8171**	.5227	.701	**.9055**	.5406	.9036	.8367	0	**.8525**	.805	.6365	.8238	**.8205**	.5389
glass6	.94	**.8849**	**.9063**	.7625	.737	**.8915**	.8487	.8831	.9198	.5068	**.8474**	.835	.833	.7942	**.8229**	.8966
segment0	.9898	**.9888**	**.9944**	.8039	0	**.9337**	.6513	.9331	0	0	**.98**	.9826	.9808	.9819	**.9841**	.8054
ecoli2	.3082	**.5081**	**.869**	.5937	.78	**.5349**	.3737	.5317	.8288	0	**.8791**	.798	.762	.7755	**.8647**	.8228
new-thyroid1	.9	**.8955**	**.9187**	.9232	.9125	**.8368**	.8396	.8377	.3798	.8363	**.9621**	.9135	.943	.9086	**.9565**	.9377
new-thyroid2	.9649	**.9159**	**.9037**	.8757	.8883	**.9351**	.8194	.9351	.9245	.7813	**.9767**	.8643	.9067	.9217	**.9329**	.9166
ecoli1	.8729	**.5809**	**.7385**	.7656	.8879	**.5872**	.5063	.6207	.856	.1553	**.8622**	.8144	.8752	.8491	**.8155**	.8664
vehicle0	.9537	**.8864**	**.8054**	.6081	.1283	**.7649**	.649	.757	.1529	.0176	**.9077**	.89	.7973	.9258	**.9007**	.581
glass0-1-2-3_vs_4-5-6	.9311	**.8707**	**.8697**	.899	.8087	**.8535**	.7987	.8349	.8016	.6304	**.8857**	.7999	.7759	.8924	**.8719**	.8953
haberman	.3655	**.4787**	**.5229**	.3231	.373	**.4526**	.139	.4518	.1759	.3356	**.4526**	.05	.0989	.3563	**.5977**	.3649
vehicle3	.5982	**.6914**	**.5318**	.4377	.1102	**.6512**	.5085	.6287	.053	.1296	**.5883**	.4475	.3816	.5725	**.6521**	0
vehicle2	.9706	**.9263**	**.8509**	.7516	.3629	**.8841**	.8397	.8831	.1723	.5017	**.9475**	.8814	.8677	.9423	**.9448**	.4968
vehicle1	.6406	**.6274**	**.5771**	.4469	.3492	**.5895**	.5136	.5984	0	.0356	**.5623**	.4377	.3251	.5828	**.6692**	.0842
yeast1	.4159	**.6876**	**.5708**	.4231	.4864	**.4115**	.0821	.4096	.2785	0	**.6123**	.4772	.3858	.6125	**.612**	.1427
glass0	.8329	**.3093**	**.772**	.6963	.6307	**.684**	.5651	.6932	.5951	0	**.8103**	.4917	.5605	.8102	**.7453**	.3119
iris0	1	**.9897**	**1**	1	1	**.8983**	.9116	.9065	.9897	.7246	**.9949**	.9795	.9848	.9897	**.9789**	.9897
pima	.6805	**.6914**	**.6317**	.6323	.6013	**.5038**	.3695	.5065	0	.1639	**.6564**	.6245	.6199	.6979	**.6694**	.5533
ecoli0_vs_1	.9767	**.**	**.969**	.9663	.9597	**.8773**	.7953	.892	.9761	.1265	**.9797**	.9699	.9451	.9831	**.969**	.9828
wisconsin	.9697	**.9622**	**.9661**	.9581	.9377	**.9538**	.9466	.9657	.5293	.5318	**.9523**	.9174	.9051	.9405	**.9321**	.9393
glass1	.7507	**.554**	**.7451**	.6412	.4875	**.5079**	.4304	.5314	.6569	.1976	**.7007**	.3734	.2554	.6562	**.6857**	.6438
Mean	.5893	**.6492**	**.6962**	.5613	.5273	**.7088**	.4808	.7096	.3959	.2076	**.6758**	.5748	.5176	.6987	**.7581**	.4693

Table B.5: Results for genetics-based algorithms on imbalanced datasets using the GM measure

dataset	CART	AQ	CN2	C4.5rules	RIPPER
abalone19	0	.5664	0	0	.232
yeast6	.6803	.5587	0	.7327	.7999
yeast5	.7145	.6812	.4133	.8489	.8739
yeast4	.1084	.3839	.2121	.5508	.6382
yeast2_vs_8	.5086	.5366	.5808	.2813	.7189
glass5	.939	.7186	0	.8804	.7347
abalone9_vs_18	.2514	.1979	.2078	.4755	.5337
glass4	.7637	.7423	.5855	.5822	.7889
ecoli4	.7788	.5598	.6398	.8551	.8841
glass2	.2508	.6572	0	.6008	.3666
vowel0	.8468	.6006	.4883	.9683	.9573
page-blocks0	.7104	.117	.5929	.9309	.9357
ecoli3	.6812	.4538	.5604	.6956	.8069
yeast3	.8082	.8117	.1856	.8723	.9181
glass6	.8089	.881	.8114	.7921	.8449
segment0	.986	.9283	.6388	.9839	.9882
ecoli2	.8124	.5542	.5173	.8608	.8618
new-thyroid1	.9462	.8656	.8566	.9178	.9288
new-thyroid2	.9118	.9289	.8566	.9327	.9275
ecoli1	.8291	.4212	.5593	.857	.9152
vehicle0	.9228	.6986	.2988	.9283	.9088
glass0-1-2-3_vs_4-5-6	.9101	.7651	.8219	.9098	.9053
haberman	.2544	.1385	.232	.3506	.5554
vehicle3	.5363	.4378	.3094	.6175	.7128
vehicle2	.9352	.8908	.4964	.9364	.9533
vehicle1	.5007	.4255	.3967	.6799	.7101
yeast1	.5397	.3503	.1014	.6896	.6772
glass0	.746	.6311	.2162	.783	.768
iris0	1	.9225	.9173	.9897	.9789
pima	.7025	.1477	.3862	.6895	.6966
ecoli0_vs_1	.9686	.8705	.885	.9831	.9582
wisconsin	.9223	.8536	.9484	.9481	.9638
glass1	.7336	.441	.4538	.6935	.7396
Mean	.6972	.5981	.4597	.7521	.7934

Table B.6: Results for classical algorithms on imbalanced datasets using the GM measure

dataset	Michigan approach		IRL approach		GCCL Approach				Pittsburgh approach					HEDT approach		
	XCS	UCS	**SIA**	HIDER	**CORE**	OCEC	COGIN	GIL	Pitts-GIRLA	DMEL	**GAssist**	OIGA	ILGA	**DT-GA**	Oblique-DT	Target
abalone19	**.6708**	0	**.2335**	.6458	**.6701**	.5584	.5514	.5543	0	.007	**.4818**	.6242	.4734	**.1558**	.1867	.6351
yeast6	**.8704**	.7737	**.831**	.8471	**.7762**	.6692	.514	.6027	0	0	**.8081**	.7255	.768	**.8057**	.752	.8527
yeast5	**.964**	.8404	**.9479**	.9735	**.937**	.647	.5063	.6892	0	.1384	**.9558**	.8588	.8977	**.9093**	.8914	.9304
yeast4	**.8056**	.5629	**.7236**	.6627	**.8057**	.4867	.291	.3767	.3296	.0118	**.7672**	.8171	.7236	**.6666**	.6244	.8207
yeast2_vs_8	**.6861**	.5965	**.7061**	.6174	**.7211**	.6911	.6457	.6937	0	.2305	**.7677**	.309	.3396	**.8101**	.7711	.7115
glass5	**.9198**	.1414	**.9139**	.7298	**.8663**	.9152	.8679	.7699	0	.3496	**.9147**	.468	.3305	**.8678**	.9138	.845
abalone9_vs_18	**.6927**	.3836	**.6918**	.6901	**.6945**	.5003	.4592	.4802	0	.0417	**.7024**	.5951	.6119	**.5279**	.6542	.6802
glass4	**.832**	.3405	**.8744**	.5947	**.6098**	.7892	.5857	.7348	0	.6114	**.6892**	.6263	.7875	**.875**	.8337	.8597
ecoli4	**.9065**	.6833	**.907**	.7654	**.9056**	.7463	.6405	.7459	0	.2947	**.8635**	.8685	.8097	**.7455**	.8465	.8311
glass2	**.6409**	0	**.6823**	.2602	**.6078**	.493	.6377	.6485	0	0	**.5469**	.5843	.4404	**.5136**	.6613	.5921
vowel0	**.9944**	.8905	**.9904**	.8496	**.7989**	.6564	.6154	.6017	.178	.4202	**.9645**	.9226	.8614	**.9372**	.9804	.8846
page-blocks0	**.9446**	.8577	**.9224**	.9217	**.7341**	.9217	.4804	.73	.3451	0	**.8887**	.8828	.8635	**.9486**	.9371	.8347
ecoli3	**.8379**	.5929	**.8443**	.8271	**.8286**	.5018	.4568	.4324	.1844	.1549	**.8308**	.8369	.7206	**.8143**	.7553	.8503
yeast3	**.9184**	.8935	**.8716**	.8623	**.8684**	.8041	.646	.762	0	0	**.9203**	.9014	.916	**.8877**	.8419	.8827
glass6	**.8688**	.8098	**.8711**	.8393	**.8846**	.9185	.7705	.9142	.9133	.3666	**.8946**	.9188	.8559	**.8902**	.8873	.8794
segment0	**.9862**	.9906	**.9919**	.9233	**.7304**	.9392	.8977	.921	0	.1411	**.9898**	.9853	.9896	**.9911**	.9891	.9408
ecoli2	**.8929**	.7909	**.91**	.9043	**.8739**	.5877	.4272	.534	.8791	.0265	**.8977**	.83	.8602	**.8725**	.8938	.823
new-thyroid1	**.9741**	.9796	**.983**	.9656	**.9346**	.8697	.7541	.8535	.3826	.7838	**.9484**	.9647	.934	**.9606**	.9803	.9355
new-thyroid2	**.9657**	.9623	**.986**	.9628	**.896**	.8523	.781	.8491	.1309	.7953	**.9572**	.9329	.9567	**.9649**	.9685	.9344
ecoli1	**.8904**	.4842	**.8438**	.857	**.8944**	.3715	.3739	.4104	0	.149	**.8525**	.86	.8704	**.9001**	.8201	.8775
vehicle0	**.9399**	.9254	**.8452**	.8573	**.7547**	.9045	.7442	.7612	.3326	.1242	**.9442**	.9162	.9031	**.9086**	.9259	.7824
glass0-1-2-3_vs_4-5-6	**.8948**	.7613	**.9348**	.8316	**.8603**	.7971	.6279	.8092	.8804	.6518	**.9313**	.8571	.8709	**.9235**	.9177	.9242
haberman	**.5647**	.4549	**.5086**	.6385	**.6593**	.535	.373	.384	.3604	.1671	**.6069**	.5209	.4097	**.6159**	.5079	.5434
vehicle3	**.7418**	.6802	**.6666**	.6793	**.1955**	.6917	.6055	.6454	0	.1578	**.7367**	.6595	.653	**.6685**	.6804	.6514
vehicle2	**.9688**	.943	**.7741**	.8336	**.3973**	.906	.7359	.7546	0	.5746	**.9524**	.9097	.9071	**.9429**	.9391	.7622
vehicle1	**.7404**	.653	**.6471**	.6924	**.6271**	.664	.6268	.6104	.2936	.0713	**.7628**	.6227	.6426	**.6785**	.7104	.6229
yeast1	**.6974**	.6428	**.6395**	.6622	**.6585**	.4712	.253	.3233	.3163	0	**.7273**	.6856	.6798	**.7076**	.6788	.6599
glass0	**.813**	.3439	**.8392**	.7054	**.697**	.7101	.6242	.71	0	0	**.8427**	.7666	.7745	**.779**	.7536	.6571
iris0	**1**	1	**.9949**	.9949	**1**	.9491	.9331	.9382	0	.7424	**1**	.9847	.9696	**.9897**	.9789	1
pima	**.7027**	.7149	**.6708**	.6813	**.7036**	.5599	.3637	.4683	.3031	.0934	**.7398**	.6978	.7056	**.7097**	.666	.6899
ecoli0_vs_1	**.9763**	.9735	**.9515**	.9017	**.9797**	.9726	.8565	.8697	.9595	.0516	**.9794**	.9767	.9657	**.9831**	.969	.951
wisconsin	**.9673**	.959	**.9605**	.9634	**.9472**	.9505	.9492	.9576	.1908	.3715	**.951**	.9159	.9221	**.948**	.9384	.948
glass1	**.7553**	.584	**.8233**	.6845	**.6243**	.6301	.4665	.5073	0	.3543	**.8007**	.6846	.6491	**.7003**	.7131	.6116
Mean	**.8492**	.673	**.8179**	.7826	**.7619**	.717	.6079	.668	.2115	.2389	**.8369**	.7791	.7595	**.8061**	.8051	.8002

Table B.7: Results for genetics-based algorithms on SMOTE-preprocessed imbalanced datasets using the GM measure

dataset	CART	AQ	CN2	C4.5rules	RIPPER
abalone19	0	.1826	.5541	.0799	.388
yeast6	.6803	.4271	.6022	.8031	.7325
yeast5	.7145	.3416	.6816	.9507	.8471
yeast4	.1084	.2871	.2259	.7583	.6567
yeast2_vs_8	.5086	.4169	.6944	.7418	.5875
glass5	.939	.9061	.7189	.9114	.7777
abalone9_vs_18	.2514	.0708	.4971	.6646	.738
glass4	.7637	.7109	.7696	.68	.6894
ecoli4	.7788	.5561	.7459	.8428	.8239
glass2	.2508	.6266	.6512	.6804	.6233
vowel0	.8468	.5536	.4986	.9659	.9747
page-blocks0	.7104	.0064	.7343	.9455	.9436
ecoli3	.6812	.3998	.1999	.8183	.8716
yeast3	.8082	.7514	.764	.9355	.8826
glass6	.8089	.6039	.8659	.8958	.9495
segment0	.986	.9196	.7802	.9891	.9914
ecoli2	.8124	.5191	.4977	.8864	.8406
new-thyroid1	.9462	.8325	.8535	.9485	.9434
new-thyroid2	.9118	.8327	.8612	.977	.9711
ecoli1	.8291	.4079	.4214	.8826	.8509
vehicle0	.9228	.7529	.763	.9225	.9352
glass0-1-2-3_vs_4-5-6	.9101	.5514	.8057	.8711	.8553
haberman	.4707	.0298	.3958	.669	.3478
vehicle3	.5363	.4124	.5238	.7213	.6652
vehicle2	.9352	.7149	.6812	.9546	.967
vehicle1	.5007	.3535	.5674	.6918	.6485
yeast1	.5397	.2493	.2793	.6994	.6792
glass0	.746	.5884	.7223	.7576	.7952
iris0	1	.9534	.9434	.9897	.9789
pima	.7596	.0878	.479	.7047	.6944
ecoli0_vs_1	.9686	.8665	.8665	.9758	.9513
wisconsin	.9223	.9455	.9398	.9558	.9441
glass1	.7336	.4725	.4077	.7204	.6837
Mean	.7055	.5252	.6361	.8179	.7948

Table B.8: Results for classical algorithms on SMOTE-preprocessed imbalanced datasets using the GM measure

APPENDIX B. RESULTS FROM THE REFERENCE WORK FOR GENETICS-BASED AND CLASSICAL ALGORITHMS

Appendix C

N_S values for different *coverage* values for datasets from the KEEL repository

This appendix contains the full tables containing the number of samples or *N_S* values for the 96 datasets and 10 *coverage* values (plus a static value of 3) used in the experiments of Section 3.6.1.

Data set	Original			Training sample			Maj. Class Size	Subsample	t_{ifc}	Coverage *sizeOfMinClass* *N_S*							
	Size	#Class	%Min	Size	Min. Class Size	MinCover[1]		Size		NS=3	10%	20%	30%	40%	50%	75%	90%	95%	99%	99.9%
nursery	1296	5	0.08%	1037	1	11	346	55	3%	3	7	12	16	22	43	72	93	143	214	
abalone	418	22	0.24%	335	1	4	56	88	7.15%	3	3	4	5	7	10	19	32	41	63	94
ecoli	336	8	0.60%	269	2	3	115	24	2.61%	3	4	9	14	20	27	53	88	114	175	262
lymphography	148	4	1.36%	119	2	2	66	8	3.04%	3	4	8	12	17	23	46	75	98	150	225
car	1728	4	3.77%	1383	53	14	969	108	2.79%	3	4	8	13	19	25	50	82	107	163	245
zoo	101	7	3.97%	81	4	1	33	14	6.07%	3	3	4	6	9	12	23	37	48	74	111
flare	1066	6	4.04%	853	35	9	265	108	6.80%	3	3	4	6	8	10	20	33	43	66	99
glass	214	6	4.21%	172	8	2	62	24	6.46%	3	3	4	6	8	11	21	35	45	70	104
cleveland	297	5	4.38%	238	11	3	129	30	4.66%	3	3	5	8	11	15	30	49	63	97	146
dermatology	358	6	5.59%	287	17	3	89	54	10.12%	3	3	3	4	5	7	14	22	29	44	65
balance	625	3	7.84%	500	40	5	231	60	8.66%	3	3	3	4	6	8	16	26	34	51	77
penbased	1100	10	9.55%	880	84	9	92	420	45.66%	3	3	3	3	3	3	3	4	5	8	12
newthyroid	215	3	13.96%	172	24	2	120	36	10%	3	3	3	4	5	7	14	22	29	44	66
hepatitis	80	2	16.25%	64	11	1	54	12	11.12%	3	3	3	4	5	6	12	20	26	40	59
contraceptive	1473	3	22.61%	1179	267	12	504	402	26.59%	3	3	3	3	3	3	5	8	10	15	23
vehicle	846	4	23.53%	677	160	7	175	320	45.72%	3	3	3	3	3	3	3	4	5	8	12
haberman	306	2	26.48%	245	65	3	181	66	18.24%	3	3	3	3	3	4	7	12	15	23	35
wine	178	3	26.97%	143	39	2	58	60	34.49%	3	3	3	3	3	4	6	8	8	11	17
breast	277	2	29.25%	222	65	3	158	66	20.89%	3	3	3	3	3	6	10	13	20	30	
german	1000	2	30%	800	240	8	560	240	21.43%	3	3	3	3	3	6	10	13	20	29	
iris	150	3	33.34%	120	40	2	40	60	50%	3	3	3	3	3	3	4	5	7	10	
wisconsin	630	2	34.61%	504	175	6	330	176	26.67%	3	3	3	3	3	5	8	10	15	23	
tictactoe	958	2	34.66%	767	266	8	502	266	26.50%	3	3	3	3	3	5	8	10	15	23	
pima	768	2	34.90%	615	215	7	401	216	26.94%	3	3	3	3	3	5	8	10	15	23	
magic	1902	2	35.13%	1522	535	16	988	536	27.13%	3	3	3	3	3	5	8	10	15	22	
bupa	345	2	42.03%	276	116	3	160	116	36.25%	3	3	3	3	3	4	6	7	11	16	
heart	270	2	44.45%	216	96	3	120	96	40%	3	3	3	3	3	5	6	10	14		
australian	690	2	44.5%	552	246	6	307	246	40.07%	3	3	3	3	3	5	9	14			
crx	653	2	45.33%	523	238	6	286	238	41.61%	3	3	3	3	3	5	9	13			
ring	740	2	49.60%	592	294	6	299	294	49.17%	3	3	3	3	3	5	9	13			
Mean	638.94	4.27	22%	511.44	111.67	5.57	256.54	147.97	22%	3	4	4	5	8	15	24	31	47	70	
StdDev	493.55	3.96	16%	394.87	126.76	3.92	245.53	139.38	53%	0.18	0.77	2.22	3.71	5.68	9.46	17.84	26.94	37.25	56.61	72.13
Median	521.5	3	23.07%	417.5	59	4.5	167.5	92	21.16%	3	3	3	3	3	3	6	10	13	20	29.5

Table C.1: *sizeOfMinClass* subsample numbers for standard datasets.

Data set	Original			Training sample				Subsample	t_{1fc}	Coverage *maxSize*										
	Size	#Class	%Min	Size	Min. Class Size	MinCover[1]	Maj. Class Size	Size		*N_S*										
										NS=3	10%	20%	30%	40%	50%	75%	90%	95%	99%	99.9%
nursery	1296	5	0.08%	1037	1	11	346	105	6.07%	3	3	4	6	9	12	23	37	48	74	111
abalone	418	22	0.24%	335	1	4	56	154	12.50%	3	3	3	3	4	6	11	18	23	35	52
ecoli	336	8	0.60%	269	2	3	115	48	5.22%	3	3	5	7	10	13	26	43	56	86	129
lymphography	148	4	1.36%	119	2	2	66	12	4.55%	3	3	5	8	11	15	30	50	65	99	149
car	1728	4	3.77%	1383	53	14	969	212	5.47%	3	3	4	7	10	13	25	41	54	82	123
zoo	101	7	3.97%	81	4	1	33	28	12.13%	3	3	3	3	4	6	11	18	24	36	54
flare	1066	6	4.04%	853	35	9	265	210	13.21%	3	3	3	3	4	5	10	17	22	33	49
glass	214	6	4.21%	172	8	2	62	48	12.91%	3	3	3	3	4	6	11	17	22	34	51
cleveland	297	5	4.38%	238	11	3	129	55	8.53%	3	3	3	5	6	8	16	26	34	52	78
dermatology	358	6	5.59%	287	17	3	89	102	19.11%	3	3	3	3	3	4	7	11	15	22	33
balance	625	3	7.84%	500	40	5	231	120	17.32%	3	3	3	3	3	4	8	13	16	25	37
penbased	1100	10	9.55%	880	84	9	92	840	91.31%	3	3	3	3	3	3	3	3	3	3	3
newthyroid	215	3	13.96%	172	24	2	120	72	20%	3	3	3	3	3	4	7	11	14	21	31
hepatitis	80	2	16.25%	64	11	1	54	22	20.38%	3	3	3	3	3	4	7	11	14	21	31
contraceptive	1473	3	22.61%	1179	267	12	504	801	52.98%	3	3	3	3	3	3	3	4	4	7	10
vehicle	846	4	23.53%	677	160	7	175	640	91.43%	3	3	3	3	3	3	3	3	3	3	3
haberman	306	2	26.48%	245	65	3	181	130	35.92%	3	3	3	3	3	3	4	6	7	11	16
wine	178	3	26.97%	143	39	2	58	117	67.25%	3	3	3	3	3	3	3	3	3	5	7
breast	277	2	29.25%	222	65	3	158	130	41.14%	3	3	3	3	3	3	3	5	6	9	14
german	1000	2	30%	800	240	8	560	480	42.86%	3	3	3	3	3	3	3	5	6	9	13
iris[3]	150	3	33.34%	120	40	2	40	66	55.00%	3	3	3	3	3	3	3	3	4	6	9
wisconsin	630	2	34.61%	504	175	6	330	350	53.04%	3	3	3	3	3	3	3	4	4	7	10
tictactoe	958	2	34.66%	767	266	8	502	532	52.99%	3	3	3	3	3	3	3	4	4	7	10
pima	768	2	34.90%	615	215	7	401	430	53.62%	3	3	3	3	3	3	3	3	4	6	9
magic	1902	2	35.13%	1522	535	16	988	1070	54.15%	3	3	3	3	3	3	3	3	4	6	9
bupa	345	2	42.03%	276	116	3	160	232	72.50%	3	3	3	3	3	3	3	3	3	4	6
heart	270	2	44.45%	216	96	3	120	192	80%	3	3	3	3	3	3	3	3	3	3	5
australian	690	2	44.50%	552	246	6	307	492	80.14%	3	3	3	3	3	3	3	3	3	3	5
crx	653	2	45.33%	523	238	6	286	476	83.22%	3	3	3	3	3	3	3	3	3	3	4
ring	740	2	49.60%	592	294	6	299	588	98.33%	3	3	3	3	3	3	3	3	3	3	3
Mean	638.94	4.27	22%	511.44	111.67	5.57	256.54	291.8	43.00%	3	3	4	4	5	6	9	13	16	24	36
StdDev	493.55	3.96	16%	394.87	126.76	3.92	245.53	280.88	31.00%	0	0	0.55	1.43	2.42	3.53	7.96	13.65	18.08	27.82	41.93
Median	521.5	3	23.07%	417.5	59	4.5	167.5	173	42.00%	3	3	3	3	3	3	3	5	6	9	13.5

Table C.2: *maxSize* subsample numbers for standard datasets.

Table C.3: *sizeOfMinClass* subsample numbers for imbalanced datasets.

Data set	Original		Training sample				Subsample	t_{1fc}	Coverage *sizeOfMinClass* *N_S*									
	Size	%Min	Size	Min. Class Size	Maj. Class Size		Size		NS=3	10%	20%	30%	40%	50%	75%	90%	95%	99%	99.9%
abalone19	4174	0.77%	3340	26	3314		26	0.39%	3	27	57	91	130	177	353	586	763	1172	1758
yeast6	1484	2.49%	1188	30	1158		30	1.30%	3	9	18	28	40	54	107	177	230	354	530
yeast5	1484	2.96%	1189	36	1153		36	1.56%	3	7	15	23	33	45	89	147	191	293	440
yeast4	1484	3.43%	1188	41	1147		41	1.79%	3	6	13	20	29	39	77	128	167	256	384
yeast2_vs_8	482	4.15%	387	17	370		17	2.30%	3	5	10	16	22	30	60	100	129	199	298
glass5	214	4.2%	173	8	165		8	2.42%	3	5	10	15	21	29	57	94	123	188	282
abalone9_vs_18	731	5.65%	586	34	552		34	3.08%	3	4	8	12	17	23	45	74	96	148	221
glass4	214	6.07%	172	11	161		11	3.42%	3	4	7	11	15	20	40	67	87	133	199
ecoli4	336	6.74%	270	19	251		19	3.78%	3	3	6	10	14	18	36	60	78	120	180
glass2	214	8.78%	173	16	157		16	5.10%	3	3	5	7	10	14	27	45	58	89	133
vowel0	988	9.01%	792	72	720		72	5%	3	3	5	7	10	14	28	45	59	90	135
page-blocks0	5472	10.23%	4378	448	3930		448	5.70%	3	3	4	7	9	12	24	40	52	79	118
ecoli3	336	10.88%	270	30	240		30	6.25%	3	3	4	6	8	11	22	36	47	72	108
yeast3	1484	10.98%	1188	131	1057		131	6.20%	3	3	4	6	8	11	22	36	47	72	108
glass6	214	13.55%	173	24	149		24	8.05%	3	3	3	5	7	9	17	28	36	55	83
segment0	2308	14.26%	1848	264	1584		264	8.33%	3	3	3	5	6	8	16	27	35	53	80
ecoli2	336	15.48%	270	42	228		42	9.21%	3	3	3	4	6	8	15	24	32	48	72
new-thyroid1	215	16.28%	173	29	144		29	10.07%	3	3	3	4	5	7	14	22	29	44	66
new-thyroid2	215	16.89%	173	30	143		30	10.49%	3	3	3	4	5	7	13	21	28	42	63
ecoli1	336	22.92%	270	62	208		62	14.90%	3	3	3	4	5	7	13	21	28	42	63
vehicle0	846	23.64%	677	160	517		160	15.47%	3	3	3	4	5	9	15	22	29	43	
glass0-1-2-3_vs_4-5-6	214	23.83%	172	41	131		41	15.65%	3	3	3	4	5	9	14	18	28	42	
haberman	306	27.42%	246	68	178		68	19.10%	3	3	3	4	5	7	11	15	22	33	
vehicle1	846	28.37%	678	193	485		193	19.90%	3	3	3	4	5	7	11	14	21	32	
vehicle2	846	28.37%	678	193	485		193	19.90%	3	3	3	4	5	7	11	14	21	32	
vehicle3	846	28.37%	678	193	485		193	19.90%	3	3	3	4	5	7	11	14	21	32	
yeast1	1484	28.91%	1188	344	844		344	20.38%	3	3	3	4	4	7	11	14	21	31	
glass0	214	32.71%	172	56	116		56	24.14%	3	3	3	3	3	6	9	11	17	26	
iris0	150	33.33%	121	40	81		40	24.69%	3	3	3	3	3	5	9	11	17	25	
pima	768	34.84%	616	215	401		215	26.81%	3	3	3	3	3	5	8	10	15	23	
ecoli_0vs_1	220	35%	177	62	115		62	26.96%	3	3	3	3	3	5	8	10	15	22	
wisconsin	683	35%	548	192	356		192	26.97%	3	3	3	3	3	5	8	10	15	22	
glass1	214	35.51%	172	61	111		61	27.48%	3	3	3	3	3	5	8	10	15	22	
Mean	919.94	17.61%	737.09	96.61	640.48		96.61	12.02%	3	4	7	10	13	18	35	58	75	115	172
StdDev	1151.99	11.71%	921.47	105.74	863.98		105.74	9.00%	0	4.3	9.81	15.93	23	31.41	62.77	104.29	135.81	208.72	313.07
Median	482	15.48%	387	42	356		42	9.21%	3	3	3	4	6	8	15	24	32	48	72

Data set	Original		Training sample			Subsample	t_{lfc}	Coverage *sizeOfMinClass* *N_S*										
	Size	%Min	Size	Min. Class Size	Maj. Class Size	Size		NS=3	10%	20%	30%	40%	50%	75%	90%	95%	99%	99.9%
abalone19	4174	0.77%	3340	26	3314	52	0.78%	3	14	29	46	65	89	177	293	381	585	878
yeast6	1484	2.49%	1188	30	1158	60	2.59%	3	5	9	14	20	27	53	88	115	176	264
yeast5	1484	2.96%	1189	36	1153	72	3.12%	3	4	8	12	17	22	44	73	95	146	218
yeast4	1484	3.43%	1188	41	1147	82	3.57%	3	3	7	10	15	20	39	64	83	127	190
yeast2_vs_8	482	4.15%	387	17	370	34	4.59%	3	3	5	8	11	15	30	49	64	98	147
blass5	214	4.2%	173	8	165	16	4.85%	3	3	5	8	11	14	28	47	61	93	139
abalone9_vs_18	731	5.65%	586	34	552	68	6.16%	3	3	4	6	9	11	22	37	48	73	109
glass4	214	6.07%	172	11	161	22	6.83%	3	3	4	6	8	10	20	33	43	66	98
ecoli4	336	6.74%	270	19	251	38	7.57%	3	3	3	5	7	9	18	30	39	59	88
glass2	214	8.78%	173	16	157	32	10.19%	3	3	3	4	5	7	13	22	28	43	65
vowel0	988	9.01%	792	72	720	144	10%	3	3	3	4	5	7	14	22	29	44	66
page-blocks0	5472	10.23%	4378	448	3930	896	11.40%	3	3	3	3	5	6	12	20	25	39	58
ecoli3	336	10.88%	270	30	240	60	12.50%	3	3	3	3	4	6	11	18	23	35	52
yeast3	1484	10.98%	1188	131	1057	262	12.39%	3	3	3	3	4	6	11	18	23	35	53
glass6	214	13.55%	173	24	149	48	16.11%	3	3	3	3	3	4	8	14	18	27	40
segment0	2308	14.26%	1848	264	1584	528	16.67%	3	3	3	3	3	4	8	13	17	26	38
ecoli2	336	15.48%	270	42	228	84	18.42%	3	3	3	3	3	4	7	12	15	23	34
new-thyroid1	215	16.28%	173	29	144	58	20.14%	3	3	3	3	3	4	7	11	14	21	31
new-thyroid2	215	16.89%	173	30	143	60	20.98%	3	3	3	3	3	3	6	10	13	20	30
ecoli1	336	22.92%	270	62	208	124	29.81%	3	3	3	3	3	3	4	7	9	14	20
vehicle0	846	23.64%	677	160	517	320	30.95%	3	3	3	3	3	3	4	7	9	13	19
glass0-1-2-3_vs_4-5-6	214	23.83%	172	41	131	82	31.30%	3	3	3	3	3	3	4	7	8	13	19
Haberman	306	27.42%	246	68	178	136	38.20%	3	3	3	3	3	3	3	5	7	10	15
vehicle1	846	28.37%	678	193	485	386	39.79%	3	3	3	3	3	3	3	5	6	10	14
vehicle2	846	28.37%	678	193	485	386	39.79%	3	3	3	3	3	3	3	5	6	10	14
vehicle3	846	28.37%	678	193	485	386	39.79%	3	3	3	3	3	3	3	5	6	10	14
yeast1	1484	28.91%	1188	344	844	688	40.76%	3	3	3	3	3	3	3	5	6	9	14
glass0	214	32.71%	172	56	116	112	48.28%	3	3	3	3	3	3	3	4	5	7	11
iris0	150	33.33%	121	40	81	80	49.38%	3	3	3	3	3	3	3	4	5	7	11
pima	768	34.84%	616	215	401	430	53.62%	3	3	3	3	3	3	3	3	4	6	9
ecoli0_vs_1	220	35%	177	62	115	124	53.91%	3	3	3	3	3	3	3	3	4	6	9
wisconsin	683	35%	548	192	356	384	53.93%	3	3	3	3	3	3	3	3	4	6	9
glass1	214	35.51%	172	61	111	122	54.95%	3	3	3	3	3	3	3	3	4	6	9
Mean	919.94	17.61%	737.09	96.61	640.48	193.21	24.04%	3	3	4	6	7	9	17	28	37	56	84
StdDev	1151.99	11.71%	921.47	105.74	863.98	211.47	18.00%	0	1.94	4.66	7.76	11.27	15.56	31.47	52.23	68	104.38	156.68
Median	482	15.48%	387	42	356	84	18.42%	3	3	3	3	3	4	7	12	15	23	34

Table C.4: *maxSize* subsample numbers for imbalanced datasets.

APPENDIX C. *N_S* VALUES FOR DIFFERENT *COVERAGE* VALUES FOR DATASETS FROM THE KEEL REPOSITORY

Appendix D

Full result tables for CTC45 using two sample sizes and eleven *coverage* values

This appendix contains the full result tables related to the summary tables shown on Section 3.6.1. These tables show the kappa and accuracy values (for standard classification) and GM values (for imbalanced classification) for all datasets and all *coverage* values.

Numbers in bold indicate the best average value for each dataset.

dataset	coverage values										
	NS=3	10%	20%	30%	40%	50%	75%	90%	95%	99%	99.9%
nursery	.8131	.8202	.812	.819	.8204	.8212	.8204	.8245	.8257	.8289	**.8342**
abalone	**.0922**	**.0922**	.0878	.0839	.0831	.0694	.07	.0698	.0627	.0647	.0715
ecoli	.6705	.6774	.6946	.6855	**.7027**	.6871	.6951	.6971	.6947	.6942	.6808
lymphography	.4819	.468	.4742	.5132	.5202	.5239	.5335	.5249	**.5644**	.55	.5124
car	.6393	.6667	.6522	.6712	.6824	.6896	.7263	.7414	.7464	.7515	**.7543**
zoo	.9137	.9137	**.9162**	.9054	.9157	.9076	.9133	.9132	.9076	.9076	.9076
flare	.6657	.6657	.6702	.6698	.668	**.6717**	.6697	.6692	.6695	.6681	.6697
glass	.5662	.5662	.5631	**.5793**	.561	.5686	.5561	.5293	.5582	.5645	.5704
cleveland	**.2546**	**.2546**	.2373	.2217	.2347	.2253	.2304	.2405	.2374	.2472	.2418
dermatology	.9134	.9134	.9134	.9141	.914	**.9266**	.9183	.9176	.9162	.9218	.921
balance	.6086	.6086	.6086	.6109	**.6156**	.6074	.6132	.6059	.6049	.6042	.598
penbased	.8834	.8834	.8834	.8834	.8834	.8834	.8834	**.885**	.8838	.8769	.8765
newthyroid	.8472	.8472	.8472	.8539	.8449	.8537	.8583	.8722	**.8832**	.8778	.8755
hepatitis	.1956	.1956	.1956	.2557	.3472	.3593	.2933	.317	.3488	**.4015**	.3812
contraceptive	.2669	.2669	.2669	.2669	.2669	.2669	.2532	.2701	**.2701**	.2608	.2696
vehicle	.6264	.6264	.6264	.6264	.6264	.6264	.6264	.6333	.6418	**.6424**	.6405
haberman	.1553	.1553	.1553	.1553	.1553	.1283	.1328	.1212	.1458	.1259	**.1567**
wine	.9049	.9049	.9049	.9049	.9049	.9049	.8982	.9119	**.9273**	.8956	.9219
breast	.2216	.2216	.2216	.2216	.2216	.2216	.2476	.2469	**.256**	.2523	.2519
german	.2949	.2949	.2949	.2949	.2949	.2949	**.2959**	.2892	.2891	.2788	.2873
iris	.904	.904	.904	.904	.904	.904	.904	**.906**	.896	.894	.898
wisconsin	.8336	.8336	.8336	.8336	.8336	.8336	.8428	.848	**.8502**	.8469	.8489
tictactoe	.6452	.6452	.6452	.6452	.6452	.6452	.6625	.6649	**.6732**	.6671	.6707
pima	**.4028**	**.4028**	**.4028**	**.4028**	**.4028**	**.4028**	.3981	.3831	.3842	.366	.39
magic	**.5149**	**.5149**	**.5149**	**.5149**	**.5149**	**.5149**	.494	.5012	.5023	.4887	.4747
bupa	.2925	.2925	.2925	.2925	.2925	.2925	.3319	**.3377**	.3364	.3367	.3191
heart	.5619	.5619	.5619	.5619	.5619	.5619	.5619	**.5743**	.5603	.5736	.5558
australian	.6926	.6926	.6926	.6926	.6926	.6926	.6926	.6903	.6876	**.6938**	.688
crx	.7099	.7099	.7099	.7099	.7099	.7099	.7099	.6968	.7088	**.7121**	.7078
ring	.7113	.7113	.7113	.7113	.7113	.7113	.7113	.7097	.7189	**.7205**	.7097
Mean	.5761	.5771	.5765	.5802	.5844	.5836	.5848	.5864	**.5917**	.5905	.5895
Median	.6329	.6358	.6358	.6358	.6358	.6358	.6445	.6491	**.6556**	.6548	.6551

Table D.1: Results for kappa over standard datasets using *sizeOfMinClass* subsamples.

dataset	*coverage* values										
	NS=3	10%	20%	30%	40%	50%	75%	90%	95%	99%	99.9%
nursery	.8111	.8111	.8183	.8236	.8203	.8276	.8307	.8312	.8339	.8352	**.8362**
abalone	.081	.081	.081	.081	.0804	.0748	**.0819**	.073	.0633	.0676	.0682
ecoli	.6972	.6972	.7004	.6964	.6926	.6863	.698	.6949	.6918	.6917	**.7024**
lymphography	.4774	.4774	.5203	.5233	.5688	.5767	.5756	.5558	.5937	.5977	**.6028**
car	.7037	.7037	.7136	.7317	.7303	.7441	.7455	.7492	**.7528**	.7524	.7458
zoo	.9212	.9212	.9212	.9212	.9266	**.9268**	.9183	.9104	.9104	.9132	.9132
flare	.6674	.6674	.6674	.6674	**.6694**	.6663	.6652	.6676	.6659	.6654	.6649
glass	.543	.543	.543	.543	**.556**	.5531	.5377	.5504	.5551	.534	.5313
cleveland	.2438	.2438	.2438	.227	**.2525**	.2364	.2308	.2229	.2229	.2309	.2246
dermatology	**.9211**	**.9211**	**.9211**	**.9211**	**.9211**	.9204	.9064	.9092	.9112	.9113	.9078
balance	.6276	.6276	.6276	.6276	.6276	**.6339**	.6023	.6013	.6047	.604	.5968
penbased	**.8802**	**.8802**	**.8802**	**.8802**	**.8802**	**.8802**	**.8802**	**.8802**	**.8802**	**.8802**	**.8802**
newthyroid	.8805	.8805	.8805	.8805	.8805	.8875	.8888	.8933	.8918	.884	**.8957**
hepatitis	.3008	.3008	.3008	.3008	.3008	.3678	.4413	.4556	.4546	**.472**	.4528
contraceptive	.2556	.2556	.2556	.2556	.2556	.2556	.2556	**.2688**	**.2688**	.2574	.2488
vehicle	**.6377**	**.6377**	**.6377**	**.6377**	**.6377**	**.6377**	**.6377**	**.6377**	**.6377**	**.6377**	**.6377**
haberman	.1363	.1363	.1363	.1363	.1363	.1363	.1343	.1195	.1201	.1372	**.1377**
wine	.9171	.9171	.9171	.9171	.9171	.9171	.9171	.9171	.9171	**.9291**	.9257
breast	.2668	.2668	.2668	.2668	.2668	.2668	.2668	.2777	.2723	.2806	**.2875**
german	.2708	.2708	.2708	.2708	.2708	.2708	.2708	.29	.2929	**.3179**	.3096
iris	.898	.898	.898	.898	.898	.898	.898	.898	.902	.904	**.906**
wisconsin	.8448	.8448	.8448	.8448	.8448	.8448	.8448	.8456	.8456	.8445	**.8531**
tictactoe	.6905	.6905	.6905	.6905	.6905	.6905	.6905	.688	.688	**.7008**	.6991
pima	.3859	.3859	.3859	.3859	.3859	.3859	.3859	.3859	.3798	**.3998**	.3732
magic	**.4908**	**.4908**	**.4908**	**.4908**	**.4908**	**.4908**	**.4908**	**.4908**	.4852	.4699	.4732
bupa	**.3095**	**.3095**	**.3095**	**.3095**	**.3095**	**.3095**	**.3095**	**.3095**	**.3095**	.2923	.285
heart	**.456**	**.456**	**.456**	**.456**	**.456**	**.456**	**.456**	**.456**	**.456**	**.456**	.4359
australian	**.6893**	**.6893**	**.6893**	**.6893**	**.6893**	**.6893**	**.6893**	**.6893**	**.6893**	**.6893**	.6843
crx	.6961	.6961	.6961	.6961	.6961	.6961	.6961	.6961	.6961	.6961	**.7182**
ring	**.7087**	**.7087**	**.7087**	**.7087**	**.7087**	**.7087**	**.7087**	**.7087**	**.7087**	**.7087**	**.7087**
Mean	.5803	.5803	.5824	.5826	.5854	.5879	.5885	.5891	.59	**.592**	.5902
Median	.6525	.6525	.6525	.6525	**.6535**	.652	.6515	.6526	.6518	.6515	.6513

Table D.2: Results for kappa over standard datasets using *maxSize* subsamples.

dataset	*coverage* values										
	NS=3	10%	20%	30%	40%	50%	75%	90%	95%	99%	99.9%
nursery	.8737	.8783	.8726	.877	.878	.8787	.8777	.8806	.881	.8838	**.8873**
abalone	**.183**	**.183**	.1788	.1761	.1763	.1633	.1628	.1629	.1568	.1582	.1641
ecoli	.7598	.766	.7809	.7741	**.7853**	.7736	.7801	.781	.7798	.779	.7695
lymphography	.7265	.7181	.7307	.7481	.7514	.7519	.7545	.7489	**.7679**	.7662	.7395
car	.8324	.8442	.8365	.8465	.851	.8547	.8711	.8786	.8814	.884	**.8852**
zoo	.9345	.9345	.9365	.9292	**.9369**	.9312	.9349	.9349	.9312	.9312	.9312
flare	.7414	.7414	.7447	.7447	.7436	**.7461**	.7448	.7446	.7448	.7437	.7447
glass	.6807	.6807	.6816	**.6904**	.6797	.6842	.673	.6533	.6735	.6789	.6841
cleveland	**.5441**	**.5441**	.5338	.5189	.5256	.523	.5262	.5284	.5259	.5332	.5251
dermatology	.9314	.9314	.9314	.9314	.9315	**.94**	.9353	.9348	.9336	.9378	.936
balance	.7848	.7848	.7848	.785	**.7857**	.781	.7839	.7773	.7763	.7746	.7703
penbased	.8948	.8948	.8948	.8948	.8948	.8948	.8948	**.8961**	.8945	.8888	.8883
newthyroid	.9276	.9276	.9276	.9319	.9278	.9309	.9327	.9386	**.9429**	.9403	.9378
hepatitis	.8313	.8313	.8313	.8433	.8604	**.8608**	.8329	.8417	.8479	.8583	.8508
contraceptive	.5276	.5276	.5276	.5276	.5276	.5276	.5173	.5283	.5279	.5219	**.5284**
vehicle	.7206	.7206	.7206	.7206	.7206	.7206	.7206	.7254	**.7338**	.7322	.7311
haberman	**.7193**	**.7193**	**.7193**	**.7193**	**.7193**	.7112	.7103	.7052	.7127	.7043	.7097
wine	.9386	.9386	.9386	.9386	.9386	.9386	.9322	.9437	**.9532**	.9328	.9495
breast	.7299	.7299	.7299	.7299	.7299	.7299	.7426	.745	.7458	**.7468**	.7457
german	.722	.722	.722	.722	.722	.722	**.7224**	.7199	.7186	.7173	.7189
iris	.9362	.9362	.9362	.9362	.9362	.9362	.9362	**.9376**	.9307	.9293	.932
wisconsin	.9237	.9237	.9237	.9237	.9237	.9237	.9279	.9304	**.9313**	.9298	.9307
tictactoe	.8447	.8447	.8447	.8447	.8447	.8447	.8526	.8548	**.8557**	.8532	.8547
pima	**.7351**	**.7351**	**.7351**	**.7351**	**.7351**	**.7351**	.7296	.7236	.7233	.7147	.7261
magic	**.7871**	**.7871**	**.7871**	**.7871**	**.7871**	**.7871**	.7742	.7778	.7793	.7721	.7658
bupa	.662	.662	.662	.662	.662	.662	.6781	.6781	**.6798**	.679	.6733
heart	.7844	.7844	.7844	.7844	.7844	.7844	.7844	.7873	.7816	**.7878**	.7801
australian	.8479	.8479	.8479	.8479	.8479	.8479	.8479	.8467	.8455	**.849**	.8461
crx	.8542	.8542	.8542	.8542	.8542	.8542	.8542	.8486	.8537	**.8559**	.8536
ring	.8571	.8571	.8571	.8571	.8571	.8571	.8571	.8567	**.8615**	.8608	.8553
Mean	.7746	.775	.7752	.7761	.7773	.7765	.7764	.777	**.7791**	.7782	.7772
Median	.786	.786	.786	.7861	**.7864**	.7858	.7842	.7841	.7807	.7834	.7752

Table D.3: Results for accuracy over standard datasets using *sizeOfMinClass* subsamples.

dataset	*coverage* values										
	NS=3	10%	20%	30%	40%	50%	75%	90%	95%	99%	99.9%
nursery	.8722	.8722	.877	.8806	.8783	.8832	.8852	.8855	.8874	.8883	**.8889**
abalone	.1733	.1733	.1733	.1733	**.1743**	.169	.1743	.168	.1585	.1618	.1623
ecoli	.781	.781	.7834	.7811	.7786	.7732	.7816	.7792	.7768	.7774	**.7845**
lymphography	.7244	.7244	.7473	.7448	.7728	.7755	.7755	.7646	.7845	.786	**.7876**
car	.8614	.8614	.866	.8745	.8742	.8806	.8814	.8828	**.8846**	.884	.8808
zoo	.9405	.9405	.9405	.9405	**.9445**	**.9445**	.9385	.9325	.9325	.9345	.9345
flare	.742	.742	.742	.742	**.7435**	.7411	.7401	.7422	.7409	.7405	.7401
glass	.6629	.6629	.6629	.6629	.6741	.6722	.662	.6705	**.6743**	.6584	.6566
cleveland	.5291	.5291	.5291	.5138	**.53**	.5199	.5167	.5118	.507	.5104	.5071
dermatology	**.9368**	**.9368**	**.9368**	**.9368**	**.9368**	.9363	.9251	.9273	.929	.929	.9262
balance	.7926	.7926	.7926	.7926	.7926	**.7965**	.7757	.7754	.7763	.7757	.7715
penbased	**.8922**	**.8922**	**.8922**	**.8922**	**.8922**	**.8922**	**.8922**	**.8922**	**.8922**	**.8922**	**.8922**
newthyroid	.9433	.9433	.9433	.9433	.9433	.947	.947	.9498	.9488	.9451	**.9507**
hepatitis	.845	.845	.845	.845	.845	.8475	.855	.8625	.8675	**.8775**	**.8775**
contraceptive	.5189	.5189	.5189	.5189	.5189	.5189	.5189	**.5279**	**.5279**	.52	.5147
vehicle	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**
haberman	**.7129**	**.7129**	**.7129**	**.7129**	**.7129**	**.7129**	.7064	.6972	.6972	.7103	.7063
wine	.9455	.9455	.9455	.9455	.9455	.9455	.9455	.9455	.9455	**.9535**	.9513
breast	.7488	.7488	.7488	.7488	.7488	.7488	.7488	.7523	.7494	.7544	**.7559**
german	.7148	.7148	.7148	.7148	.7148	.7148	.7148	.719	.7186	**.73**	.7256
iris	.932	.932	.932	.932	.932	.932	.932	.932	.9347	.936	**.9373**
wisconsin	.9302	.9302	.9302	.9302	.9302	.9302	.9302	.9305	.9305	.9302	**.934**
tictactoe	.8641	.8641	.8641	.8641	.8641	.8641	.8641	.8637	.8637	**.8687**	.8685
pima	.7247	.7247	.7247	.7247	.7247	.7247	.7247	.7247	.7198	**.7313**	.718
magic	**.7724**	**.7724**	**.7724**	**.7724**	**.7724**	**.7724**	**.7724**	**.7724**	.7703	.7627	.7633
bupa	**.6655**	**.6655**	**.6655**	**.6655**	**.6655**	**.6655**	**.6655**	**.6655**	**.6655**	.6591	.6533
heart	**.7319**	**.7319**	**.7319**	**.7319**	**.7319**	**.7319**	**.7319**	**.7319**	**.7319**	**.7319**	.7222
australian	**.8467**	**.8467**	**.8467**	**.8467**	**.8467**	**.8467**	**.8467**	**.8467**	**.8467**	**.8467**	.8443
crx	.8497	.8497	.8497	.8497	.8497	.8497	.8497	.8497	.8497	.8497	**.8604**
ring	**.8543**	**.8543**	**.8543**	**.8543**	**.8543**	**.8543**	**.8543**	**.8543**	**.8543**	**.8543**	**.8543**
Mean	.7746	.7746	.7757	.7755	.7774	.7773	.7761	.7762	.7765	**.7776**	.7766
Median	.7868	.7868	**.788**	.7869	.7856	.786	.7786	.7773	.7807	.7817	.7861

Table D.4: Results for accuracy over standard datasets using *maxSize* subsamples.

dataset	*coverage* values										
	NS=3	10%	20%	30%	40%	50%	75%	90%	95%	99%	99.9%
abalone19	.5201	**.5738**	.5623	.5325	.5119	.4554	.4065	.4291	.3762	.3944	.4335
yeast6	.8125	.8246	**.8357**	.817	.8211	.8338	.8303	.7627	.73	.7619	.7123
yeast5	**.9526**	.9444	.9238	.9087	.919	.9255	.8913	.8799	.8662	.8549	.8614
yeast4	.7897	.7804	**.8081**	.7716	.7496	.7718	.6299	.5877	.6143	.5577	.4563
yeast2_vs_8	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**
glass5	.942	.9435	.9505	.9525	.9352	.9352	.944	.9622	.9764	**.9769**	.9429
abalone9_vs_18	.6922	.6918	**.7299**	.713	.7105	.6991	.7133	.7064	.7122	.6906	.6944
glass4	.7438	.7857	.7987	.7537	.7739	.7448	.7842	.79	.7778	.7823	**.8215**
ecoli4	.8006	.8006	.8297	.8313	.8164	.8218	.7317	.8108	.823	**.8469**	.8092
glass2	**.6642**	**.6642**	.6058	.5499	.4801	.5731	.5902	.6214	.5802	.5198	.5695
vowel0	.887	.887	.9001	.8915	.8849	.9	.9154	.9147	.9175	.929	**.9293**
page-blocks0	.8881	.8881	.8965	.9041	.906	.9099	**.9175**	.9087	.9094	.907	.9052
ecoli3	.6883	.6883	.7404	.7679	.7462	.7698	.7451	.7411	.744	**.7741**	.7647
yeast3	.8838	.8838	**.8851**	.8673	.8707	.8664	.8673	.8666	.8668	.8601	.8593
glass6	.872	.872	.872	**.9011**	.8772	.8892	.876	.8526	.8407	.8358	.83
segment0	.9783	.9783	.9783	.9835	.984	.9818	.983	.9837	.9857	.9861	**.9891**
ecoli2	.8569	.8569	.8569	.859	.8602	.8666	**.8672**	.845	.8556	.8668	.8647
new-thyroid1	.9411	.9411	.9411	.9499	.9499	.9437	.9493	.9514	**.9566**	.9518	.9524
new-thyroid2	.9239	.9239	.9239	.9401	.9459	.9487	.9575	.9568	**.9598**	.952	.9485
ecoli1	.8546	.8546	.8546	.8546	.8583	.8564	.8566	.8433	**.8591**	.8502	.8415
vehicle0	.9229	.9229	.9229	.9229	.9255	**.9297**	.9196	.9242	.9187	.9206	.9215
glass0-1-2-3_vs_4-5-6	.8797	.8797	.8797	.8797	.8724	**.8885**	.8639	.8708	.864	.8686	.8566
haberman	**.5205**	**.5205**	**.5205**	**.5205**	**.5205**	.4977	.4891	.4434	.4862	.4627	.4629
vehicle1	.5996	.5996	.5996	.5996	.5996	.6129	.6326	.6279	**.6592**	.6378	.6376
vehicle2	.9326	.9326	.9326	.9326	.9326	.9321	.9347	.9358	.9341	.9338	**.9439**
vehicle3	.5806	.5806	.5806	.5806	.5806	.6241	.6292	.6479	.6454	**.6529**	.6515
yeast1	.6479	.6479	.6479	.6479	.6479	.6397	.6327	.6378	**.6521**	.6275	.6416
glass0	.7668	.7668	.7668	.7668	.7668	.7668	.7852	.7811	.8006	.7947	**.8078**
iris0	.9815	.9815	.9815	.9815	.9815	.9815	**.9836**	**.9836**	**.9836**	**.9836**	.9815
pima	.6734	.6734	.6734	.6734	.6734	.6734	.6908	.6841	**.6912**	.691	.6821
ecoli0_vs_1	.9802	.9802	.9802	.9802	.9802	.9802	.9802	**.9809**	.9774	.9795	.9795
wisconsin	**.9293**	**.9293**	**.9293**	**.9293**	**.9293**	**.9293**	.9262	.9278	.9229	.9192	.9171
glass1	.6648	.6648	.6648	.6648	.6648	.6648	.6756	.707	**.7246**	.6966	.6828
Mean	.803	.8058	**.8091**	.8048	.8001	.8043	.7978	.7968	.7982	.7938	.7903
Median	.8546	.8546	.8546	.8546	**.8583**	.8564	.8566	.8433	.8407	.8469	.83

Table D.5: Results for GM over imbalanced datasets using *sizeOfMinClass* subsamples.

dataset	*coverage* values										
	NS=3	10%	20%	30%	40%	50%	75%	90%	95%	99%	99.9%
abalone19	**.5823**	.4918	.5103	.4752	.4726	.4093	.454	.3878	.4111	.4383	.4024
yeast6	.7911	.814	.8174	.8091	.8161	**.8345**	.7874	.757	.7747	.7203	.7389
yeast5	.9303	**.9443**	.9288	.9316	.9086	.9003	.8913	.8693	.8705	.8648	.8742
yeast4	.7487	.7487	**.7544**	.7222	.6549	.6221	.5709	.5761	.5237	.5193	.5279
yeast2_vs_8	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**	**.7283**
glass5	.9158	.9158	.9249	.9304	.977	**.9801**	.9764	.9555	.9437	.9437	.9437
abalone9_vs_18	.7135	.7135	**.7283**	.719	.7007	.7009	.6559	.6888	.6593	.6453	.6678
glass4	.8549	.8549	.8481	.864	.8453	.8542	**.864**	.8438	.8347	.8184	.8139
ecoli4	**.8599**	**.8599**	**.8599**	.8362	.808	.8082	.8188	.8326	.8249	.8321	.8353
glass2	**.6882**	**.6882**	**.6882**	.6608	.6399	.656	.5709	.6431	.6708	.624	.5618
vowel0	.9115	.9115	.9115	.9128	.9107	.931	.9277	**.9351**	.9335	.9286	.9289
page-blocks0	.919	.919	.919	.919	.9232	**.9242**	.9217	.9202	.916	.9167	.9143
ecoli3	.7509	.7509	.7509	.7509	.7641	.7648	.7436	.7608	**.7783**	.7472	.7335
yeast3	.8852	.8852	.8852	.8852	**.8865**	.868	.8745	.8538	.8624	.8594	.8638
glass6	.8679	.8679	.8679	.8679	.8679	**.8708**	.8394	.8356	.8435	.8586	.8444
segment0	.9853	.9853	.9853	.9853	.9853	.9866	.9881	.9881	.9893	**.9896**	.9884
ecoli2	.859	.859	.859	.859	.859	.8692	**.8768**	.8674	.8657	.8764	.8683
new-thyroid1	.9577	.9577	.9577	.9577	.9577	**.965**	.9608	.9583	.948	.9571	.9509
new-thyroid2	.9488	.9488	.9488	.9488	.9488	.9488	.9573	.9567	.9568	**.9643**	.9632
ecoli1	.8629	.8629	.8629	.8629	.8629	.8629	**.8752**	.859	.8498	.8623	.8612
vehicle0	.9184	.9184	.9184	.9184	.9184	.9184	.9302	.929	**.9324**	.929	.9301
glass0-1-2-3_vs_4-5-6	.877	.877	.877	.877	.877	.877	.8764	**.8783**	.8707	.8606	.8765
haberman	**.5908**	**.5908**	**.5908**	**.5908**	**.5908**	**.5908**	**.5908**	.5273	.5337	.5323	.5053
vehicle1	.6578	.6578	.6578	.6578	.6578	.6578	.6578	**.6628**	.6582	.6537	.6597
vehicle2	.9395	.9395	.9395	.9395	.9395	.9395	.9395	.9445	.9466	**.9497**	.9488
vehicle3	.6536	.6536	.6536	.6536	.6536	.6536	.6536	.6575	.6622	.6628	**.6648**
yeast1	.6424	.6424	.6424	.6424	.6424	.6424	.6424	.6498	**.6563**	.6545	.6482
glass0	.785	.785	.785	.785	.785	.785	.785	.797	**.802**	.7714	.7914
iris0	**.9897**	**.9897**	**.9897**	**.9897**	**.9897**	**.9897**	**.9897**	**.9897**	**.9897**	**.9897**	**.9897**
pima	**.6923**	**.6923**	**.6923**	**.6923**	**.6923**	**.6923**	**.6923**	**.6923**	.6902	.6849	.6848
ecoli0_vs_1	**.9823**	**.9823**	**.9823**	**.9823**	**.9823**	**.9823**	**.9823**	**.9823**	.9815	.9816	.9816
wisconsin	.923	.923	.923	.923	.923	.923	.923	.923	**.928**	.926	.9251
glass1	.6953	.6953	.6953	.6953	.6953	.6953	.6953	.6953	**.7082**	.706	.6909
Mean	**.8215**	.8198	.8207	.8174	.8141	.8131	.8073	.8044	.8044	.7999	.7972
Median	.8599	.8599	.8599	.8629	.859	.8629	**.864**	.8438	.8435	.8586	.8444

Table D.6: Results for GM over imbalanced datasets using *maxSize* subsamples.

dataset	*coverage* values										
	NS=3	10%	20%	30%	40%	50%	75%	90%	95%	99%	99.9%
abalone19	.6164	.6229	.5733	.5429	**.6244**	.583	.5317	.4337	.4703	.4754	.431
yeast6	**.8697**	.8642	.8391	.8358	.8552	.8273	.8586	.8363	.851	.8502	.856
yeast5	.9437	.9398	.9553	.9659	.9674	.9651	.9629	.9619	.9693	**.9733**	.9717
yeast4	.8091	.8092	.8027	.8044	.8137	.8029	.8027	.8081	**.8182**	.7893	.7665
yeast2_vs_8	.7455	.751	.7532	.7588	.7454	.771	.7643	.7961	**.8137**	.793	.802
glass5	.8725	.9321	.9391	**.9433**	.9407	.887	.9357	.8546	.8646	.8589	.9272
abalone9_vs_18	.6833	**.7011**	.6619	.6366	.6596	.6726	.6703	.6556	.6544	.6173	.6082
glass4	.7955	.752	.7914	.8104	.7899	.82	.8228	**.8342**	.8044	.7836	.8263
ecoli4	**.8631**	**.8631**	.858	.8248	.8227	.8526	.8459	.8143	.8255	.8409	.8522
glass2	.5539	.5539	.5605	.5195	.606	.6215	.4801	**.6514**	.5478	.6436	.5542
vowel0	.9286	.9286	.9296	.9296	.9336	.9297	.9333	.9374	.9399	**.9419**	.932
page-blocks0	.9358	.9358	.9355	.9368	.9377	.942	.9421	.9422	.9418	**.9441**	.9441
ecoli3	.8552	.8552	.8559	.8569	.8608	**.881**	.8647	.8667	.8665	.8693	.8548
yeast3	.9187	.9187	.917	.9203	.9182	.9164	.9143	.918	.9158	**.9219**	.9211
glass6	.896	.896	.896	**.9025**	.8973	.8981	.8898	.8764	.8739	.8795	.8701
segment0	.9784	.9784	.9784	.9832	.9834	.9857	.9857	.9885	.9891	.9895	**.9906**
ecoli2	.8545	.8545	.8545	.8404	.865	.8543	.8618	.8655	.8559	**.8658**	.8646
new-thyroid1	.9169	.9169	.9169	.919	.9249	.9345	.9403	**.9548**	.9503	.9535	.945
new-thyroid2	.9079	.9079	.9079	.8969	.9251	.9241	.9342	.9371	.9392	**.9507**	.9493
ecoli1	.8921	.8921	.8921	.8921	.8904	.8867	.8853	.8903	.8851	**.8943**	.892
vehicle0	.9327	.9327	.9327	.9327	.937	**.9373**	.9356	.9323	.9335	.9308	.9352
glass0-1-2-3_vs_4-5-6	.8776	.8776	.8776	.8776	.8757	.8824	**.8953**	.8848	.8926	.8807	.8896
haberman	.6064	.6064	.6064	.6064	.6064	.611	.6135	.6038	**.6185**	.6116	.616
vehicle1	.7204	.7204	.7204	.7204	.7204	.7133	.7107	**.7371**	.7225	.7146	.7087
vehicle2	.9341	.9341	.9341	.9341	.9341	.9352	.9368	.9334	.9406	.9406	**.9452**
vehicle3	.7226	.7226	.7226	.7226	.7226	.7364	**.7404**	.7147	.7272	.7132	.7211
yeast1	.6964	.6964	.6964	.6964	.6964	.6981	.7058	.7029	.7021	**.7109**	.71
glass0	.7846	.7846	.7846	.7846	.7846	.7846	.7904	.7949	.7871	**.7999**	.7951
iris0	**.9815**	**.9815**	**.9815**	**.9815**	**.9815**	**.9815**	**.9815**	**.9815**	.9784	**.9815**	**.9815**
pima	.7076	.7076	.7076	.7076	.7076	.7076	.7155	.7178	.71	**.7222**	.7177
ecoli0_vs_1	.9779	.9779	.9779	.9779	.9779	.9779	.9737	.9737	.9774	.976	**.9781**
wisconsin	.9347	.9347	.9347	.9347	.9347	.9347	.9398	.9415	.9416	.9404	**.9434**
glass1	.6966	.6966	.6966	.6966	.6966	.6966	.7048	.7045	**.7239**	.7118	.7231
Mean	.8306	.8317	.83	.8271	.8344	**.8349**	.8324	.8317	.8313	.8324	.831
Median	.8697	.8642	.858	.8569	.865	**.881**	.8647	.8655	.8646	.8658	.8646

Table D.7: Results for GM over imbalanced datasets preprocessed with SMOTE using *sizeOfMinClass* subsamples.

dataset	*coverage* values										
	NS=3	10%	20%	30%	40%	50%	75%	90%	95%	99%	99.9%
abalone19	**.6509**	.5867	.5148	.529	.4918	.4195	.4734	.4793	.4945	.4794	.4413
yeast6	**.8506**	.837	.8405	.842	.8384	.8378	.8412	.8503	.838	.8241	.8344
yeast5	.9367	.9527	.9642	.9701	.9687	.9601	.9706	**.9728**	.9718	.9608	.9526
yeast4	.8102	.8102	.7942	.8012	.8094	**.8184**	.7799	.7617	.7539	.7527	.7281
yeast2_vs_8	.7447	.7447	.7464	.7617	.7827	.7881	.7921	.8034	.803	**.8077**	.7833
glass5	.8554	.8554	.8732	.8733	.8809	.8382	.7984	.8628	**.9227**	.9136	.9145
abalone9_vs_18	.634	.634	**.6551**	.6321	.648	.6274	.6308	.6509	.6092	.6157	.6106
glass4	.8302	.8302	.8478	.8388	.8278	.8347	.8405	**.8494**	.8451	.8377	.8221
ecoli4	.8339	.8339	.8339	.8053	.8155	.8339	.8443	.8424	.8145	.8551	**.8681**
glass2	.6504	.6504	.6504	.6831	.6781	.6911	.6993	.7046	.7167	.734	**.7578**
vowel0	.9391	.9391	.9391	.9404	.9479	.9373	.9468	.9514	.942	.9508	**.9578**
page-blocks0	.943	.943	.943	.943	.9435	.9421	.9441	.9402	.944	**.9442**	.9416
ecoli3	.8589	.8589	.8589	.8589	.8601	.8647	**.8714**	.852	.8681	.8636	.8452
yeast3	.9194	.9194	.9194	.9194	**.9221**	.9205	.9149	.9137	.9115	.9067	.9054
glass6	.8904	.8904	.8904	.8904	.8904	**.8927**	.8795	.845	.8501	.8545	.8637
segment0	.9873	.9873	.9873	.9873	.9873	.9871	.988	.9882	.9891	.9888	**.9894**
ecoli2	.8531	.8531	.8531	.8531	.8531	**.8657**	.8567	.846	.8531	.8472	.8453
new-thyroid1	.9445	.9445	.9445	.9445	.9445	.9538	.9525	**.9595**	.9568	.9562	.9498
new-thyroid2	.9455	.9455	.9455	.9455	.9455	.9455	.9461	.9462	.9474	**.9565**	.9542
ecoli1	.8951	.8951	.8951	.8951	.8951	.8951	.8911	.887	.893	.8884	**.8953**
vehicle0	.9289	.9289	.9289	.9289	.9289	.9289	.9288	**.9359**	.932	.9296	.9301
glass0-1-2-3_vs_4-5-6	.881	.881	.881	.881	.881	.881	.8689	.8826	.8939	.8881	**.895**
haberman	**.6149**	**.6149**	**.6149**	**.6149**	**.6149**	**.6149**	**.6149**	.6089	.6005	.5977	.6085
vehicle1	.7127	.7127	.7127	.7127	.7127	.7127	.7127	.7098	.7113	**.7229**	.7016
vehicle2	.9381	.9381	.9381	.9381	.9381	.9381	.9381	.9397	.9433	**.9473**	.9465
vehicle3	**.7415**	**.7415**	**.7415**	**.7415**	**.7415**	**.7415**	**.7415**	.7208	.7131	.7174	.7211
yeast1	**.6978**	**.6978**	**.6978**	**.6978**	**.6978**	**.6978**	**.6978**	.6924	.6957	.6928	.6856
glass0	.7915	.7915	.7915	.7915	.7915	.7915	.7915	.8025	.7886	.8032	**.8147**
iris0	.9856	.9856	.9856	.9856	.9856	.9856	.9856	.9856	**.9877**	**.9877**	**.9877**
pima	.6957	.6957	.6957	.6957	.6957	.6957	.6957	.6957	.7036	**.704**	.6974
ecoli0_vs_1	.9773	.9773	.9773	.9773	.9773	.9773	.9773	.9773	**.9787**	.9779	.9743
wisconsin	.9394	.9394	.9394	.9394	.9394	.9394	.9394	.9394	.9419	.9435	**.9438**
glass1	**.7186**	**.7186**	**.7186**	**.7186**	**.7186**	**.7186**	**.7186**	**.7186**	.7111	.7176	.7118
Mean	**.8363**	.8344	.8339	.8345	.835	.8326	.8325	.8338	.8341	.8354	.8327
Median	.8554	.8554	.8589	.8589	.8601	**.8647**	.8567	.8503	.8531	.8551	.8637

Table D.8: Results for GM over imbalanced datasets preprocessed with SMOTE using *maxSize* subsamples.

APPENDIX D. FULL RESULT TABLES FOR CTC45 USING TWO SAMPLE SIZES AND ELEVEN *COVERAGE* VALUES

Appendix E

Analysis of the effect of *coverage*-based resampling on the true-positive and true-negative rates for imbalanced datasets

In one of the contributions of this thesis, when analyzing the effect of different *coverage* values on the performance of the CTC45 algorithm on imbalanced datasets (See Section 3.6.1), the results show that whereas the kappa and accuracy increase with *coverage* on standard datasets, the GM on imbalanced datasets decreases.

In Figure E.1 and Figure E.2 the same results are shown for maxSize subsamples, however, this time the individual rates that conform the GM are also shown.

It can be observed that in both cases (with and without the SMOTE preprocessing), the decrease in GM is due to a decrease in the true-positive rate, as the true-negative does indeed increase, albeit at a lower rate than the true-positve decrease, which turns into a decrease for the geometric mean. This is expected, using balanced subsamples, each added sample brings more new information for the negative class than for the positive class. In fact, using *maxSize* samples on non-preprocessed datasets, each and every sample already has all available information about the minority class, whereas in the case of the majority class, each new sample has the potential of containing all new information.

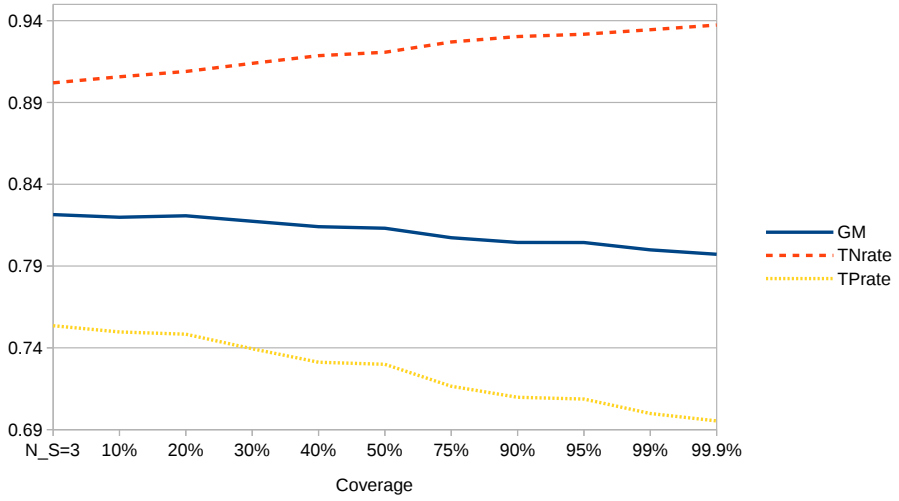

Figure E.1: Performance of CTC45 for different *coverage* values with *maxSize* subsamples on imbalanced datasets.

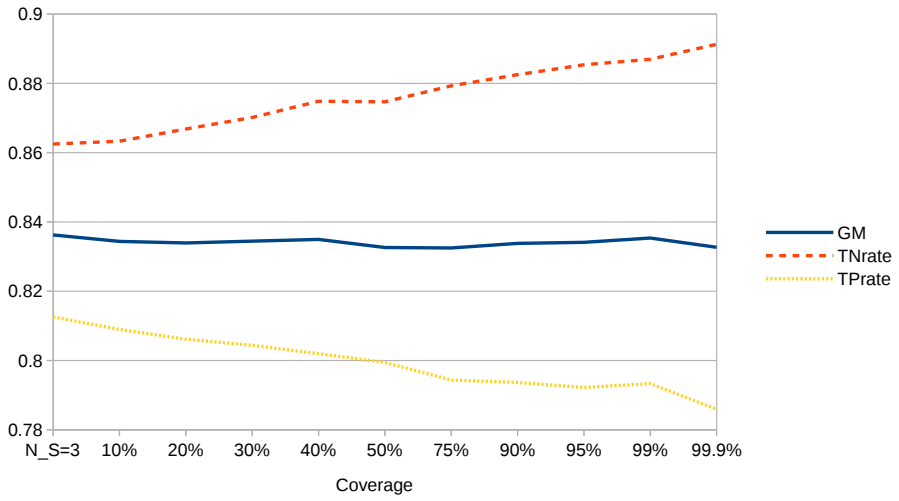

Figure E.2: Performance of CTC45 for different *coverage* values with *maxSize* subsamples on imbalanced datasets preprocessed with SMOTE.

Appendix F

Full result tables for the second experiment from the contributions to the consolidation of decision tree algorithms

This appendix contains the full result tables related to the summary tables shown on Section 3.6.2. These tables show the kappa and accuracy values (for standard classification) and GM values (for imbalanced classification) for all datasets and all GBML, classical, and consolidated algorithms.

Numbers in bold indicate the best average value for each dataset.

	genetics-based				classical					base decision trees			consolidated decision trees			
dataset	XCS	SIA	OCCEC	GAssist	Oblique-DT	CART	AQ	CN2	C4.5rules	RIPPER	C4.5	C4.4	CHAID*	CHAIC	CTC45	CTC44	CTCHAID	CTCHAIC
nursery	.6988	.8467	.7353	.8553	**.8962**	.7764	.6853	.6903	.6903	.8386	.8341	.8303	.8684	.8797	.8171	.8073	.7577	.7033
abalone	.0701	.0966	.0701	.1255	.0674	.0918	.0590	**.1590**	**.1590**	.0935	.0962	.0686	.1510	.1464	.0924	.0887	.1264	.0764
ecoli	**.7043**	.6099	.5508	.5918	.6732	.6367	.5349	.5816	.5816	.6559	.7030	.6906	.6883	.6888	.6926	.6630	.5643	.5025
lymphography	.6044	.6099	.5508	.5918	.4597	.5611	.5040	**.6325**	**.6325**	.5627	.5367	.4513	.5408	.5386	.5133	.4858	.1415	.1070
car	.4321	.8550	.5535	.8238	**.9647**	.5534	.6689	.3230	.3230	.7591	.7986	.8091	.9047	.9034	.6483	.7413	.8297	.7941
zoo	.8527	.9377	.9166	.9167	**.9477**	.8815	.8855	.8197	.8197	.8828	.9215	.9086	.9365	.9365	.8826	.8771	.8516	.6932
flare	.3931	.3479	.6290	.6631	.6561	.6702	.5532	.5596	.5596	.5891	.6676	.6509	**.6861**	.6832	.6667	.6101	.6672	.6203
glass	.5350	**.6404**	.3376	.4517	.5546	.5738	.4477	.4360	.4360	.5288	.5494	.5315	.5164	.5164	.5496	.5291	.4865	.3548
cleveland	.2794	.1720	**.3067**	.2448	.2213	.1655	.1288	.0918	.0918	.2068	.2257	.2384	.2345	.2532	.2614	.2073	.2329	.2155
dermatology	.9446	.7890	.7543	.9033	.9227	.6567	.8341	.8448	.8448	.8513	.9045	.8940	.9313	.9138	.9342	.9180	**.9531**	.9510
balance	.6851	.6790	.5113	.6031	**.8347**	.3024	.2160	.6069	.6069	.3178	.5922	.6238	.6075	.6036	.6159	.5536	.5695	.4682
penbased	.8731	**.9525**	.6415	.6201	.9016	.5768	.5756	.6318	.6318	.8412	.8838	.8676	.8515	.8454	.8775	.8759	.8462	.8452
newthyroid	.8833	.8454	.7132	.8460	**.8871**	.8807	.6981	.7877	.7877	.8769	.8519	.8283	.8251	.8067	.8533	.8532	.8207	.7585
hepatitis	**.5389**	.1283	.3646	.4230	.4171	.3124	.3916	.2169	.2169	.3191	.1115	.2536	.2625	.2390	.2791	.2177	.3062	.2703
contraceptive	.2761	.1853	.1943	.2806	.1909	.2517	.0105	.0652	.0652	.2723	.2845	.2221	**.3092**	.3020	.2704	.2195	.2984	.2914
vehicle	**.6361**	.4983	.3647	.5529	.6043	.5400	.4252	.4083	.4083	.6104	.6185	.6232	.5967	.6058	.6289	.6219	.5882	.5905
haberman	.0835	.1304	.0862	.0880	.0701	.0387	.0018	.0051	.0051	.1432	.1521	.1466	.0853	.0853	.1464	.1501	**.2497**	.2479
wine	**.9520**	.9287	.5876	.8885	.8802	.8800	.6885	.7847	.7847	.8504	.9222	.9222	.9053	.9053	.9055	.9055	.8766	.8720
breast	.2781	.2191	.2082	**.3099**	.1051	.0000	.2166	.1417	.1417	.1716	.2330	.1401	.2970	.2970	.2418	.0737	.2490	.2930
german	**.3082**	.1104	.2400	.2953	.2268	.2591	.1858	.1254	.1254	.2510	.3049	.2790	.3038	.2979	.2748	.2224	.2901	.2921
iris	.9160	.9160	.8220	**.9380**	.8920	.9300	.7220	.8600	.8600	.8960	.9000	.9000	.8900	.8800	.9060	.8960	.9080	.9000
wisconsin	.9258	**.9335**	.9072	.9020	.8462	.8509	.6903	.8984	.8984	.9122	.8515	.7666	.8705	.8593	.8480	.7735	.8482	.8512
tictactoe	.6487	**.9949**	.5909	.8891	.7796	.4222	.8613	.1881	.1881	.9375	.6770	.6987	.7626	.7432	.6631	.6928	.7504	.7049
pima	.4233	.3564	.2303	.3896	.3540	.4002	.0862	.1239	.1239	.3795	.4175	.4169	**.4434**	.4370	.3887	.3448	.4032	.3968
magic	**.5643**	.4462	.1623	.5430	.4384	.4480	.1092	.1524	.1524	.4990	.5183	.5128	.4996	.4933	.4983	.4640	.5126	.4816
bupa	.2662	.2560	-.0047	.2667	.2409	.1184	.0538	.0538	.0538	.2839	.3124	.3144	.2444	.2077	.3222	.2845	.2274	.2392
heart	.5644	.3379	.5583	**.5948**	.4738	.4459	.5577	.5097	.5097	.5017	.5636	.5168	.5489	.5484	.5605	.5472	.5431	.5383
australian	.7092	.2752	**.7162**	.7042	.6345	.6947	.3876	.5884	.5884	.6300	.6886	.6220	.6698	.6930	.6930	.6327	.6743	.6669
crx	.7146	.2879	**.7262**	.7158	.5991	.6864	.5832	.5751	.5751	.6387	.7196	.6169	.7170	.7170	.6969	.5976	.7032	.6929
ring	**.8367**	.3692	.5680	.7627	.5977	.6675	-.0076	.5907	.5907	.6286	.7135	.6918	.7786	.7937	.7189	.7102	.7763	.7705
Mean	.5866	.5237	.4840	.5953	.5779	.5167	.4240	.4484	.4484	.5643	.5851	.5679	**.5976**	.5940	.5816	.5521	.5684	.5396
Median	.6202	.4722	.5522	.6116	.6017	.5573	.4759	.5346	.5346	.5997	.6431	.6226	.6386	**.6445**	.6386	.6038	.5789	.5644

Table F.1: Results for kappa over standard datasets.

	genetics-based					classical												
											base decision trees			consolidated decision trees				
dataset	XCS	SIA	OCCEC	GAssist	Oblique-DT	CART	AQ	CN2	C4.5rules	RIPPER	C4.5	C4.4	CHAID*	CHAIC	CTC45	CTC44	CTCHAID	CTCHAIC
nursery	.7969	.8948	.8162	.9023	**.9290**	.8480	.7667	.7917	.8367	.8895	.8874	.8827	.9104	.9181	.8762	.8671	.8363	.7907
abalone	.1867	.1952	.1412	.2445	.1656	.2129	.1517	**.2680**	.2097	.1737	.1938	.1699	.2560	.2512	.1953	.1598	.2422	.1226
ecoli	**.7899**	.6948	.6126	.7667	.7619	.7382	.6858	.7145	.7770	.7437	.7858	.7769	.7739	.7739	.7781	.7507	.6839	.6273
lymphography	.7946	.7970	.7633	.7873	.7166	.7761	.7208	**.8108**	.7265	.7634	.7501	.7097	.7506	.7506	.7499	.7269	.5966	.3278
car	.7955	.9337	.7752	.9206	**.9838**	.8085	.8682	.7726	.8681	.8828	.9080	.9138	.9560	.9554	.8332	.8691	.9216	.8976
zoo	.8888	.9525	.9366	.9366	**.9605**	.9105	.9145	.8710	.9310	.9111	.9410	.9310	.9514	.9514	.9110	.9070	.8874	.7444
flare	.5317	.5201	.7054	.7399	.7317	.7448	.6593	.6660	.7015	.6713	.7420	.7298	**.7570**	.7547	.7414	.6917	.7441	.6955
glass	.6647	**.7343**	.4972	.6142	.6712	.6870	.6059	.6079	.6276	.6480	.6687	.6547	.6359	.6359	.6740	.6451	.6253	.4674
cleveland	.5441	.4956	**.5697**	.5421	.4915	.4917	.5126	.5388	.5387	.4404	.5182	.5183	.5353	.5351	.5568	.4423	.5602	.4530
dermatology	.9559	.8324	.8045	.9229	.9385	.7322	.8704	.8798	.9079	.8811	.9246	.9162	.9457	.9316	.9475	.9346	**.9626**	.9609
balance	.8269	.8240	.7005	.7856	**.9056**	.6224	.5712	.7872	.8064	.5069	.7728	.7840	.7872	.7840	.7878	.7213	.7654	.6502
penbased	.8858	**.9573**	.6775	.6584	.9115	.6200	.6185	.6691	.8720	.8571	.8955	.8809	.8664	.8609	.8898	.8884	.8616	.8607
newthyroid	.9460	.9312	.8614	.9274	**.9479**	.9442	.8428	.9070	.9023	.9433	.9302	.9163	.9163	.9070	.9302	.9284	.9153	.8726
hepatitis	**.9000**	.8400	.7900	.8575	.8325	.8250	.8725	.8625	.8125	.7975	.7875	.8125	.8375	.8125	.8325	.7750	.7486	.7125
contraceptive	.5406	.4903	.4517	.5446	.4771	.5133	.4258	.4447	.5166	.5210	.5363	.4996	**.5519**	.5458	.5297	.4869	.5457	.5276
vehicle	**.7270**	.6239	.5233	.6646	.7033	.6536	.5672	.5579	.6447	.7076	.7139	.7175	.6974	.7040	.7217	.7165	.6912	.6929
haberman	.7190	.6593	.6745	.6868	.6248	.6961	.2791	.7026	.6960	.5071	.7222	.7189	.7288	**.7288**	.7116	.6529	.7050	.6863
wine	**.9683**	.9526	.7270	.9266	.9208	.9211	.7947	.8595	.9490	.9006	.9490	.9490	.9376	.9376	.9377	.9377	.9185	.9153
breast	.7489	.6946	.6540	.7394	.6284	.7076	.7285	.7221	.7071	.6144	.7437	.6788	.7654	**.7654**	.7445	.6015	.7272	.7113
german	**.7312**	.6770	.6700	.7260	.6706	.7010	.7144	.7250	.7116	.6666	.7310	.7140	.7180	.7160	.7214	.6660	.7242	.6614
iris	.9440	.9440	.8813	**.9587**	.9280	.9533	.8147	.9067	.9400	.9307	.9333	.9333	.9267	.9200	.9373	.9307	.9387	.9333
wisconsin	.9660	**.9696**	.9581	.9555	.9306	.9327	.8333	.9547	.9503	.9596	.9333	.8984	.9420	.9370	.9317	.9010	.9314	.9327
tictactoe	.8532	**.9977**	.8121	.9511	.9008	.7265	.9400	.7056	.8601	.9716	.8581	.8685	.8938	.8847	.8528	.8606	.8920	.8599
pima	**.7466**	.7190	.6963	.7327	.7083	.7187	.6703	.6732	.7284	.7005	.7357	.7344	.7448	.7422	.7260	.6995	.7328	.7159
magic	**.8124**	.7537	.6877	.7994	.7440	.7476	.6751	.6866	.7843	.7689	.7944	.7907	.7860	.7839	.7772	.7555	.7902	.7618
bupa	.6603	.6383	.4522	.6481	.6307	**.6899**	.4458	.5942	.6417	.6359	.6725	.6667	.6406	.6203	.6719	.6499	.6325	.6209
heart	.7852	.6756	.7800	**.8007**	.7400	.7259	.7859	.7667	.7778	.7474	.7852	.7630	.7778	.7778	.7830	.7748	.7748	.7711
australian	.8559	.6530	**.8591**	.8539	.8203	.8464	.7130	.8014	.8481	.8133	.8464	.8145	.8362	.8493	.8487	.8186	.8386	.8342
crx	.8579	.6561	**.8637**	.8585	.8010	.8439	.7889	.7949	.8539	.8177	.8607	.8117	.8576	.8576	.8496	.8016	.8515	.8466
ring	**.9184**	.6862	.7846	.8814	.7989	.8338	.4922	.7959	.8492	.8141	.8568	.8459	.8892	.8968	.8595	.8551	.8881	.8851
Mean	.7781	.7465	.7042	.7778	.7658	.7391	.6777	.7280	.7659	.7396	.7793	.7667	**.7858**	.7830	.7769	.7472	.7645	.7180
Median	**.7962**	.7267	.7162	.7934	.7804	.7352	.7137	.7459	.7954	.7662	.7867	.7874	.7866	.7839	.7854	.7652	.7701	.7301

Table F.2: Results for accuracy over standard datasets.

	genetics-based				classical					base decision trees			consolidated decision trees			
dataset	UCS	SLA	OCEC	GAssist	Oblique-DT	CART	AQ	CN2	C4.5rules	RIPPER	C4.5	C4.4	CHAID*	CHAIC	CTC45	CTC44	CTCHAID	CTCHAIC
abalone19	.0000	.0000	.6172	.0000	.0753	.0000	.0000	.0000	.0000	.2320	.0000	.0000	.0000	.0000	.4383	.4352	.6156	**.6443**
yeast6	.7553	.5147	.5424	.0000	.6280	.6803	.5587	.0000	.7327	**.7999**	.7335	.7339	.3111	.3111	.7203	.7947	.7193	.7107
yeast5	.7814	.7771	.7804	.7267	.8082	.7145	.6812	.4133	.8489	.8739	.8612	.8726	.8112	.8357	.8648	**.9284**	.8353	.9168
yeast4	.5996	.2621	.5831	.1716	.5556	.1084	.3839	.2121	.5508	.6382	.4208	.2993	.2322	.2124	.5193	.7072	.6867	**.7656**
yeast2_vs_8	.5657	.1000	.7145	.6274	.6905	.5086	.5366	.5808	.2813	.7189	.1723	.0984	.7274	.7264	.7283	**.7597**	.7283	.7122
glass5	.3397	.6779	.8834	.1397	.9365	.9390	.7186	.0000	.8804	.7347	.8804	.8804	**.9876**	.9876	.9437	.9774	.9582	.9613
abalone9_vs_18	.4136	.4017	.4865	.4041	.5411	.2514	.1979	.2078	.4755	.5337	.3882	.5373	.2975	.3628	.6453	.6934	**.7108**	.7079
glass4	.3612	**.9541**	.6972	.6251	.6270	.7637	.7423	.5855	.5822	.7889	.5769	.5848	.4140	.4140	.8184	.8508	.7979	.7341
ecoli4	.6873	.6521	.7606	.7802	.8809	.7788	.5598	.6398	.8551	**.8841**	.7777	.8201	.8123	.8381	.8321	.8369	.8565	.8312
glass2	.0961	.1934	.6599	.0000	.3205	.2508	.6572	.0000	.6008	.3666	.4427	.3464	.0000	.0000	.6240	.6219	**.7058**	.6859
vowel0	.9668	**1.0000**	.6238	.9360	.9559	.8468	.6006	.4883	.9683	.9573	.9683	.9627	.9298	.9298	.9286	.9367	.9152	.9348
page-blocks0	.8504	.8659	.7514	.8087	.9007	.7104	.1170	.5029	.9309	**.9357**	.9225	.9177	.8977	.9025	.9167	.9331	.9012	.9289
ecoli3	.4629	.6169	.6329	.6711	.6715	.6812	.4538	.5604	.6956	.8069	.6773	.6960	.6629	.6516	.7472	**.8133**	.7696	.8132
yeast3	.8946	.8171	.9055	.8525	.8205	.8082	.8117	.1856	.8723	**.9181**	.8479	.8552	.8541	.8508	.8594	.8799	.8732	.8984
glass6	.8849	**.9063**	.8915	.8474	.8229	.8089	.8810	.8114	.7921	.8449	.7940	.7833	.7920	.7920	.8586	.8554	.8392	.8224
segment0	.9888	**.9944**	.9337	.9800	.9841	.9860	.9283	.6388	.9839	.9882	.9814	.9816	.9761	.9761	.9896	.9933	.9868	.9869
ecoli2	.5081	.8690	.5349	**.8791**	.8647	.8124	.5542	.5173	.8608	.8618	.8497	.8401	.7564	.7576	.8764	.8760	.8649	.8595
new-thyroid1	.8955	.9187	.8368	**.9621**	.9565	.9462	.8656	.8566	.9178	.9288	.9460	.9155	.9175	.9175	.9571	.9590	.9397	.9391
new-thyroid2	.9159	.9037	.9351	**.9767**	.9329	.9118	.9289	.8566	.9327	.9275	.9327	.9327	.9019	.9019	.9643	.9593	.9357	.9312
ecoli1	.5809	.7385	.5872	.8622	.8155	.8291	.4212	.5593	.8570	**.9152**	.8538	.8481	.8194	.8194	.8623	.8707	.8528	.8801
vehicle0	.8864	.8054	.7649	.9077	.9007	.9228	.6986	.2988	.9283	.9088	.9282	.9186	.9193	.9193	.9290	.9257	.9214	.9219
glass-0-1-2-3_vs_4-5-6	.8707	.8697	.8535	.8857	.8719	.9101	.7651	.8219	.9098	.9053	.7845	**.9274**	.7726	.8773	.8606	.8686	.9048	.9022
haberman	.4787	.5229	.4526	.4526	.5977	.2544	.1385	.2320	.3506	.5554	.3563	.3544	.2147	.2147	.5323	.5883	.6170	**.6188**
vehicle1	.6274	.5771	.5895	.5623	.6692	.5363	.4378	.3094	.6175	**.7128**	.6512	.6137	.5274	.5079	.6537	.6779	.6316	.7009
vehicle2	.9263	.8509	.8841	.9475	.9448	.9352	.8908	.4964	.9364	**.9533**	.9453	.9413	.9454	.9429	.9497	.9469	.9408	.9358
vehicle3	.6914	.5318	.6512	.5883	.6521	.5007	.4255	.3967	.6799	.7101	.6728	.6722	.5542	.5497	.6628	.6876	.5550	**.7381**
yeast1	.6876	.5708	.4115	.6123	.6120	.5397	.3503	.1014	.6896	.6772	.6276	.6502	.5889	.5934	.6545	.6581	.6457	.7044
glass0	.3093	.7720	.6840	.8103	.7453	.7460	.6311	.2162	.7830	.7680	.9210	.7568	.8882	.7726	.7714	.7529	.7791	.7766
iris0	.9897	1.0000	.8983	.9949	.9789	1.0000	.9225	.9173	.9897	.9789	.9897	.9897	.9897	.9897	.9897	.9897	.9897	.9897
pima	.6914	.6317	.5038	.6564	.6694	.7025	.1477	.3862	.6895	.6966	.7023	.7026	**.7182**	.7147	.6849	.6774	.6782	.7040
ecoli0_vs_1	.2000	.9690	.8773	.9797	.9690	.9686	.8705	.8850	.9831	.9582	.9760	**.9831**	.9831	.9831	.9816	.9688	**.9831**	.9739
wisconsin	.9622	**.9661**	.9538	.9523	.9321	.9223	.8536	.9484	.9481	.9638	.9315	.8953	.9543	.9466	.9260	.8923	.9403	.9358
glass1	.5540	**.7451**	.5079	.7007	.6857	.7336	.4410	.4538	.6935	.7396	.7198	.7102	.6470	.6050	.7060	.6948	.6278	.6652
Mean	.6492	.6962	.7088	.6758	.7581	.6972	.5081	.4597	.7521	.7934	.7349	.7280	.6910	.6910	.7999	.8185	.8093	**.8252**
Median	.6876	.7720	.6972	.7802	.8155	.7637	.6006	.4883	.8489	.8449	.7940	.8201	.7920	.7920	**.8586**	.8554	.8392	.8312

Table F.3: Results for GM over imbalanced datasets.

	genetics-based					classical									consolidated decision trees			
											base decision trees							
dataset	XCS	SIA	CORE	GAssist	DT-GA	CART	AQ	CN2	C4.5rules	RIPPER	C4.5	C4.4	CHAID*	CHAIC	CTC45	CTC44	CTCHAID	CTCHAIC
abalone19	**.6708**	.2335	.6701	.4818	.1558	.0000	.1826	.5541	.0799	.3880	.1558	.1559	.1558	.1558	.4794	.4726	.4739	.4388
yeast6	**.8704**	.8310	.7762	.8081	.8057	.6803	.4271	.6022	.8031	.7325	.8048	.8050	.7262	.7253	.8241	.8171	.8333	.7230
yeast5	**.9640**	.9479	.9370	.9558	.9093	.7145	.3416	.6816	.9507	.8471	.9445	.9324	.9059	.8203	.9608	.9538	.9415	.8633
yeast4	.8056	.7236	**.8057**	.7672	.6666	.1084	.2871	.2259	.7583	.6567	.6521	.6527	.6236	.5635	.7527	.7413	.8018	.7269
yeast2_vs_8	.6861	.7061	.7211	.7677	.8101	.5086	.4169	.6944	.7418	.5875	.7402	.7403	.7039	.7039	.8077	**.8257**	.7546	.7493
glass5	.9198	.9139	.8663	.9147	.8678	.9390	.9061	.7189	.9114	.7777	.8720	.8720	.9316	.9316	.9136	.9180	.9507	**.9522**
abalone9_vs_18	.6927	.6918	.6945	.7024	.5279	.2514	.0708	.4971	.6646	**.7380**	.5319	.5357	.5456	.5456	.6157	.5971	.6695	.6423
glass4	.8320	.8744	.6098	.6892	**.8750**	.7637	.7109	.7696	.6800	.6894	.8443	.8443	.7966	.8045	.8377	.8340	.8268	.7224
ecoli4	.9065	**.9070**	.9056	.8635	.7455	.7788	.5561	.7459	.8428	.8239	.7410	.7410	.8484	.8221	.8551	.8477	.8009	.7636
glass2	.6409	.6823	.6078	.5469	.5136	.2508	.6266	.6512	.6804	.6233	.6500	.6535	.6572	.6572	**.7340**	.6976	.6069	.4832
vowel0	.9944	**.9994**	.7989	.9645	.9372	.8468	.5536	.4986	.9659	.9747	.9357	.9379	.9404	.9415	.9508	.9513	.9395	.8983
page-blocks0	.9446	.9224	.7341	.8887	**.9486**	.7104	.0064	.7343	.9455	.9436	.9428	.9371	.9244	.9195	.9442	.9390	.9364	.9171
ecoli3	.8379	.8443	.8286	.8308	.8143	.6812	.3998	.1999	.8183	.8716	.7442	.7625	.8300	.7858	.8636	.8484	**.8869**	.8161
yeast3	.9184	.8716	.8684	.9203	.8877	.8082	.7514	.7640	**.9355**	.8826	.8843	.8841	.9200	.9087	.9067	.8874	.9238	.8941
glass6	.8688	.8711	.8846	.8946	.8902	.8089	.6039	.8659	.8958	**.9495**	.8662	.8691	.8071	.8071	.8545	.8561	.8367	.8132
segment0	.9862	.9919	.7304	.9898	.9911	.9860	.9196	.7802	.9891	.9914	.9944	**.9944**	.9898	.9893	.9888	.9873	.9831	.9820
ecoli2	.8929	**.9100**	.8739	.8977	.8725	.8124	.5191	.4977	.8864	.8406	.8809	.8696	.8489	.8489	.8472	.8356	.8373	.8140
new-thyroid1	.9741	**.9830**	.9346	.9484	.9606	.9462	.8325	.8535	.9485	.9434	.9741	.9741	.9595	.9595	.9562	.9586	.9556	.9315
new-thyroid2	.9657	.9860	.8960	.9572	.9649	.9118	.8327	.8612	.9770	.9711	.9677	.9677	.9916	**.9916**	.9565	.9576	.9496	.9416
ecoli1	.8904	.8438	.8944	.8525	.9001	.8291	.4079	.4214	.8826	.8509	**.9125**	.9058	.8820	.8818	.8884	.8579	.8909	.8813
vehicle0	.9399	.8452	.7547	**.9442**	.9086	.9228	.7529	.7630	.9225	.9352	.9139	.9113	.9209	.9129	.9296	.9238	.9212	.9048
glass-0-1-2-3_vs_4-5-6	.8948	**.9348**	.8603	.9313	.9235	.9101	.5514	.8057	.8711	.8553	.7726	.8721	.7916	.9135	.8881	.8866	.9080	.8618
haberman	.5647	.5086	.6593	.6069	.6159	.4707	.0298	.3958	**.6690**	.3478	.6074	.5882	.5368	.5237	.5977	.5900	.5792	.5813
vehicle1	.7404	.6471	.6271	**.7628**	.6785	.5363	.4124	.5238	.7213	.6652	.6838	.6750	.7316	.7246	.7229	.6942	.7138	.6793
vehicle2	**.9688**	.7741	.3973	.9524	.9429	.9352	.7149	.6812	.9546	.9670	.9486	.9446	.9508	.9532	.9473	.9468	.9427	.9224
vehicle3	**.7418**	.6666	.1955	.7367	.6685	.5007	.3535	.5674	.6918	.6485	.7085	.6821	.7412	.7342	.7174	.7023	.7297	.6976
yeast1	.6974	.6395	.6585	.7273	.7076	.5397	.2493	.2793	.6994	.6792	.7135	.7125	.7186	.7165	.6928	.6638	.7014	.6965
glass0	.8130	.8392	.6970	.8427	.7790	.7460	.5884	.7223	.7576	.7952	.8668	.7644	.9223	.7872	.8032	.7735	.7731	.7380
iris0	1.0000	.9949	1.0000	1.0000	.9897	1.0000	.9534	.9434	.9897	.9789	.9897	.9897	.9897	.9897	.9877	.9877	.9897	.9877
pima	.7027	.6708	.7036	.7398	.7097	**.7596**	.0878	.4790	.7047	.6944	.7128	.7093	.6930	.6995	.7040	.6776	.7089	.7015
ecoli0_vs_1	.9763	.9515	.9797	.9794	**.9831**	.9686	.8665	.8665	.9758	.9513	.9725	.9725	.9831	.9831	.9779	.9659	.9816	.9774
wisconsin	**.9673**	.9605	.9472	.9510	.9480	.9223	.9455	.9398	.9558	.9441	.9454	.9550	.9492	.9477	.9435	.9486	.9463	.9513
glass1	.7553	**.8233**	.6243	.8007	.7003	.7336	.4725	.4077	.7204	.6837	.7480	.7429	.6965	.6632	.7176	.7228	.6922	.6827
Mean	**.8492**	.8179	.7619	.8369	.8061	.7055	.5252	.6361	.8179	.7948	.8068	.8047	.8065	.7974	.8354	.8263	.8299	.7981
Median	**.8904**	.8452	.7762	.8635	.8725	.7637	.5514	.6816	.8711	.8406	.8662	.8691	.8484	.8203	.8551	.8484	.8373	.8140

Table F.4: Results for GM over imbalanced datasets preprocessed with SMOTE.

APPENDIX F. FULL RESULT TABLES FOR THE SECOND EXPERIMENT FROM THE CONTRIBUTIONS TO THE CONSOLIDATION OF DECISION TREE ALGORITHMS

Appendix G

Full result tables for the third experiment from the contributions to the consolidation of decision tree algorithms

This appendix contains the full result tables related to the summary tables shown on Section 3.6.3. These tables show the kappa and accuracy values (for standard classification) and GM values (for imbalanced classification) for all datasets and different pruning strategies, using a *coverage* value of 99%.

Numbers in bold indicate the best average value for each dataset.

	base decision trees				consolidated decision trees			
dataset	pruned	NRT	unpruned	PET	pruned	NRT	unpruned	PET
nursery	.8684	.8684	.8797	**.8797**	.7577	.7577	.7534	.7033
abalone	**.1533**	.151	.1464	.1464	.1264	.1264	.0829	.0764
ecoli	.6805	.6883	**.6888**	.6888	.5643	.5643	.5133	.5025
lymphography	.5304	**.5408**	.5386	.5386	.1415	.1415	.1435	.107
car	**.9076**	.9047	.9034	.9034	.8297	.8297	.8226	.7941
zoo	**.9398**	.9365	.9365	.9365	.8516	.8516	.7854	.6932
flare	**.6893**	.6861	.6832	.6832	.6672	.6672	.622	.6203
glass	**.5391**	.5164	.5164	.5164	.4865	.4865	.3924	.3548
cleveland	.2256	.2345	.2532	**.2532**	.2329	.2329	.2248	.2155
dermatology	.9383	.9313	.9138	.9138	.9531	.9531	**.966**	.951
balance	.5996	**.6075**	.6036	.6036	.5695	.5695	.4909	.4682
penbased	.8373	**.8515**	.8454	.8454	.8462	.8462	.847	.8452
newthyroid	**.8686**	.8251	.8067	.8067	.8207	.8207	.8014	.7585
hepatitis	.1908	.2625	.286	.239	.0505	**.3062**	.2822	.2703
contraceptive	.2909	**.3092**	.302	.302	.2984	.2984	.2916	.2914
vehicle	.5902	.5967	.6015	**.6058**	.5882	.5882	.5916	.5905
haberman	.0265	.0853	.0853	.0853	.1529	**.2497**	.2448	.2479
wine	.8797	.9053	.9053	**.9053**	.8766	.8766	.8759	.872
breast	**.297**	**.297**	**.297**	.297	.249	.249	.293	.293
german	.2714	**.3038**	.2979	.2979	.2901	.2901	.2938	.2921
iris	**.92**	.89	.88	.88	.908	.908	.9109	.9
wisconsin	.841	**.8705**	.8593	.8593	.8482	.8482	.8552	.8512
tictactoe	.7605	**.7626**	.7432	.7432	.7504	.7504	.7054	.7049
pima	.4043	**.4434**	.437	.437	.4032	.4032	.397	.3968
magic	**.5266**	.4996	.4933	.4933	.5126	.5126	.4812	.4816
bupa	**.2444**	**.2444**	.2077	.2077	.2274	.2274	.2392	.2392
heart	**.5489**	**.5489**	.5484	.5484	.5431	.5431	.5383	.5383
australian	.6837	.6698	.693	**.693**	.6743	.6743	.6669	.6669
crx	**.7172**	.717	.717	.717	.7032	.7032	.6917	.6929
ring	.7482	.7786	.7867	**.7937**	.7763	.7763	.7726	.7705
Mean	.5906	**.5976**	.5952	.594	.5567	.5684	.5526	.5396

Table G.1: Results for different pruning strategies for CHAID* over standard datasets using kappa as measure.

	base decision trees				consolidated decision trees			
dataset	pruned	NRT	unpruned	PET	pruned	NRT	unpruned	PET
nursery	.9104	.9104	.9181	**.9181**	.8363	.8363	.8302	.7907
abalone	**.2584**	.256	.2512	.2512	.2422	.2422	.1323	.1226
ecoli	.7661	.7739	.7739	**.7739**	.6839	.6839	.641	.6273
lymphography	.7494	.7506	.7506	**.7506**	.5966	.5966	.3544	.3278
car	**.9571**	.956	.9554	.9554	.9216	.9216	.914	.8976
zoo	**.9572**	.9514	.9514	.9514	.8874	.8874	.8212	.7444
flare	**.7615**	.757	.7547	.7547	.7441	.7441	.697	.6955
glass	**.6572**	.6359	.6359	.6359	.6253	.6253	.5027	.4674
cleveland	.5355	.5353	.5351	.5351	**.5602**	**.5602**	.4475	.453
dermatology	.9512	.9457	.9316	.9316	.9626	.9626	**.9729**	.9609
balance	.784	**.7872**	.784	.784	.7654	.7654	.6696	.6502
penbased	.8536	**.8664**	.8609	.8609	.8616	.8616	.8624	.8607
newthyroid	**.9395**	.9163	.907	.907	.9153	.9153	.8982	.8726
hepatitis	.8275	**.8375**	.8125	.8125	.835	.7486	.7161	.7125
contraceptive	.5363	**.5519**	.5458	.5458	.5457	.5457	.5276	.5276
vehicle	.693	.6974	.7009	**.704**	.6912	.6912	.6937	.6929
haberman	.7288	.7288	.7288	.7288	**.73**	.705	.6831	.6863
wine	.9208	.9376	.9376	**.9376**	.9185	.9185	.9178	.9153
breast	.7654	.7654	.7654	**.7654**	.7272	.7272	.7113	.7113
german	.7127	.718	.716	.716	**.7242**	**.7242**	.662	.6614
iris	**.9467**	.9267	.92	.92	.9387	.9387	.9406	.9333
wisconsin	.9285	**.942**	.937	.937	.9314	.9314	.9345	.9327
tictactoe	.8929	**.8938**	.8847	.8847	.892	.892	.8601	.8599
pima	.7266	**.7448**	.7422	.7422	.7328	.7328	.7158	.7159
magic	**.7923**	.786	.7839	.7839	.7902	.7902	.7614	.7618
bupa	**.6406**	**.6406**	.6203	.6203	.6388	.6325	.6209	.6209
heart	.7778	.7778	.7778	**.7778**	.7748	.7748	.7711	.7711
australian	.842	.8362	.8493	**.8493**	.8386	.8386	.8342	.8342
crx	.8576	.8576	.8576	**.8576**	.8515	.8515	.8458	.8466
ring	.8741	.8892	.8932	**.8968**	.8881	.8881	.8862	.8851
Mean	.7848	**.7858**	.7828	.783	.7684	.7645	.7275	.718

Table G.2: Results for different pruning strategies for CHAID* over standard datasets using accuracy as measure.

	base decision trees				consolidated decision trees			
dataset	pruned	NRT	unpruned	PET	pruned	NRT	unpruned	PET
nursery	.8341	.8341	**.8381**	.8303	.8171	.8171	.8076	.8073
abalone	**.0962**	**.0962**	.0843	.0686	.0924	.0924	.0906	.0887
ecoli	**.703**	**.703**	.6989	.6906	.6926	.6926	.6613	.663
lymphography	**.5367**	**.5367**	.5269	.4513	.5133	.5133	.4861	.4858
car	.7986	.7986	**.8285**	.8091	.6483	.6483	.7427	.7413
zoo	**.9215**	**.9215**	.9084	.9086	.8826	.8826	.8771	.8771
flare	**.6676**	**.6676**	.6479	.6509	.6667	.6667	.6063	.6101
glass	.5494	.5494	.5345	.5315	**.5496**	**.5496**	.5301	.5291
cleveland	.2257	.2257	.2375	.2384	**.2614**	**.2614**	.2046	.2073
dermatology	.9045	.9045	.9011	.894	**.9342**	**.9342**	.918	.918
balance	.5922	.5922	**.6469**	.6238	.6159	.6159	.5536	.5536
penbased	**.8838**	**.8838**	.8818	.8676	.8775	.8775	.8759	.8759
newthyroid	.8519	.8519	.8531	.8283	**.8533**	**.8533**	.8532	.8532
hepatitis	.1115	.1115	.2395	.2536	**.2791**	**.2791**	.2177	.2177
contraceptive	**.2845**	**.2845**	.2517	.2221	.2704	.2704	.2195	.2195
vehicle	.6185	.6185	.6216	.6232	**.6289**	**.6289**	.6219	.6219
haberman	.1521	.1521	.1466	.1466	.1464	.1464	**.1565**	.1501
wine	.9222	.9222	.9222	**.9222**	.9055	.9055	.9055	.9055
breast	.233	.233	.1299	.1401	**.2418**	**.2418**	.0737	.0737
german	**.3049**	**.3049**	.2809	.279	.2748	.2748	.2224	.2224
iris	.9	.9	.9	.9	**.906**	**.906**	.896	.896
wisconsin	**.8515**	**.8515**	.7703	.7666	.848	.848	.7793	.7735
tictactoe	.677	.677	**.7085**	.6987	.6631	.6631	.6928	.6928
pima	.4175	.4175	**.4203**	.4169	.3887	.3887	.3448	.3448
magic	**.5183**	**.5183**	.5117	.5128	.4983	.4983	.464	.464
bupa	.3124	.3124	**.3307**	.3144	.3222	.3222	.2845	.2845
heart	**.5636**	**.5636**	.5203	.5168	.5605	.5605	.5472	.5472
australian	.6886	.6886	.6385	.622	**.693**	**.693**	.6327	.6327
crx	**.7196**	**.7196**	.6262	.6169	.6969	.6969	.5988	.5976
ring	.7135	.7135	.7108	.6918	**.7189**	**.7189**	.7102	.7102
Mean	**.5851**	**.5851**	.5773	.5679	.5816	.5816	.5525	.5521

Table G.3: Results for different pruning strategies for C4.5 over standard datasets using kappa as measure.

	base decision trees				consolidated decision trees			
dataset	pruned	NRT	unpruned	PET	pruned	NRT	unpruned	PET
nursery	.8874	.8874	**.8889**	.8827	.8762	.8762	.8674	.8671
abalone	.1938	.1938	.1843	.1699	**.1953**	**.1953**	.1622	.1598
ecoli	**.7858**	**.7858**	.7828	.7769	.7781	.7781	.7489	.7507
lymphography	**.7501**	**.7501**	.7434	.7097	.7499	.7499	.7269	.7269
car	.908	.908	**.9219**	.9138	.8332	.8332	.8697	.8691
zoo	**.941**	**.941**	.931	.931	.911	.911	.907	.907
flare	**.742**	**.742**	.727	.7298	.7414	.7414	.6885	.6917
glass	.6687	.6687	.6547	.6547	**.674**	**.674**	.645	.6451
cleveland	.5182	.5182	.5149	.5183	**.5568**	**.5568**	.4376	.4423
dermatology	.9246	.9246	.9218	.9162	**.9475**	**.9475**	.9346	.9346
balance	.7728	.7728	**.8**	.784	.7878	.7878	.7213	.7213
penbased	**.8955**	**.8955**	.8936	.8809	.8898	.8898	.8884	.8884
newthyroid	**.9302**	**.9302**	**.9302**	.9163	.9302	.9302	.9284	.9284
hepatitis	.7875	.7875	.8	.8125	**.8325**	**.8325**	.775	.775
contraceptive	**.5363**	**.5363**	.5152	.4996	.5297	.5297	.4869	.4869
vehicle	.7139	.7139	.7163	.7175	**.7217**	**.7217**	.7165	.7165
haberman	**.7222**	**.7222**	.7189	.7189	.7116	.7116	.6529	.6529
wine	.949	.949	.949	**.949**	.9377	.9377	.9377	.9377
breast	.7437	.7437	.6681	.6788	**.7445**	**.7445**	.6015	.6015
german	**.731**	**.731**	.71	.714	.7214	.7214	.666	.666
iris	.9333	.9333	.9333	.9333	**.9373**	**.9373**	.9307	.9307
wisconsin	**.9333**	**.9333**	.9	.8984	.9317	.9317	.9035	.901
tictactoe	.8581	.8581	**.8716**	.8685	.8528	.8528	.8606	.8606
pima	**.7357**	**.7357**	.7357	.7344	.726	.726	.6995	.6995
magic	**.7944**	**.7944**	.7902	.7907	.7772	.7772	.7555	.7555
bupa	.6725	.6725	**.6783**	.6667	.6719	.6719	.6499	.6499
heart	**.7852**	**.7852**	.763	.763	.783	.783	.7748	.7748
australian	.8464	.8464	.8217	.8145	**.8487**	**.8487**	.8186	.8186
crx	**.8607**	**.8607**	.8147	.8117	.8496	.8496	.8009	.8016
ring	.8568	.8568	.8554	.8459	**.8595**	**.8595**	.8551	.8551
Mean	**.7793**	**.7793**	.7712	.7667	.7769	.7769	.747	.7472

Table G.4: Results for different pruning strategies for C4.5 over standard datasets using accuracy as measure.

	base decision trees				consolidated decision trees			
dataset	pruned	NRT	unpruned	PET	pruned	NRT	unpruned	PET
abalone19	0	0	0	0	0	.6156	.6156	**.6443**
yeast6	.1302	.3111	.3106	.3111	.608	.7193	**.8232**	.7107
yeast5	.7715	.8112	.8354	.8357	.8353	.8353	**.9639**	.9168
yeast4	.3844	.2322	.2122	.2124	.0547	.6867	**.8178**	.7656
yeast2_vs_8	.7274	.7274	.7251	.7264	**.7283**	**.7283**	.7052	.7122
glass5	**.9876**	**.9876**	**.9876**	.9876	.237	.9582	.958	.9613
abalone9_vs_18	.2975	.2975	.4317	.3628	0	**.7108**	**.7108**	.7079
glass4	.303	.414	.414	.414	.2534	**.7979**	.7763	.7341
ecoli4	.8123	.8123	.8381	.8381	.8565	.8565	**.8866**	.8312
glass2	0	0	0	0	0	**.7058**	**.7058**	.6859
vowel0	.8722	.9298	.9298	.9298	.9152	.9152	**.952**	.9348
page-blocks0	.9002	.8977	.9059	.9025	.9012	.9012	**.9421**	.9289
ecoli3	.6749	.6629	.6516	.6516	.7696	.7696	**.8872**	.8132
yeast3	.867	.8541	.8508	.8508	.8732	.8732	**.9221**	.8984
glass6	.792	.792	.792	.792	.8392	.8392	**.8977**	.8224
segment0	.9829	.9761	.9761	.9761	.9868	.9868	**.9881**	.9869
ecoli2	.7564	.7564	.7564	.7576	.8649	.8649	**.8757**	.8595
new-thyroid1	.9503	.9175	.9175	.9175	.9397	.9397	**.9535**	.9391
new-thyroid2	.9355	.9019	.9192	.9019	.9357	.9357	**.9496**	.9312
ecoli1	.8221	.8194	.8194	.8194	.8528	.8528	**.8968**	.8801
vehicle0	.9023	.9186	.9221	.9193	.9214	.9214	**.9364**	.9219
glass0-1-2-3_vsvs_vs4-5-6	.8766	.7726	.7726	.8773	.9048	.9048	**.9136**	.9022
haberman	.0931	.2147	.2147	.2147	.2791	.617	**.6205**	.6188
vehicle1	.5033	.5274	.5764	.5079	.6316	.6316	**.715**	.7009
vehicle2	.9221	**.9454**	.9446	.9429	.9408	.9408	.9411	.9358
vehicle3	.5542	.5542	.5874	.5497	.555	.555	**.7452**	.7381
yeast1	.5889	.5889	.5934	.5934	.6457	.6457	**.7081**	.7044
glass0	.7726	**.8882**	.8787	.7726	.7791	.7791	.778	.7766
iris0	.9897	.9897	.9897	**.9897**	.9897	.9897	.9897	**.9897**
pima	.6953	.7182	**.7197**	.7147	.6782	.6782	.7078	.704
ecoli0_vs_1	**.9831**	**.9831**	**.9831**	.9831	**.9831**	**.9831**	.9781	.9739
wisconsin	.9411	**.9543**	**.9543**	.9466	.9403	.9403	.943	.9358
glass1	.6278	.647	.6302	.605	.6278	.6278	.6651	**.6652**
Mean	.6793	.691	.6982	.691	.6766	.8093	**.8445**	.8252

Table G.5: Results for different pruning strategies for CHAID* over imbalanced datasets using GM as measure.

	base decision trees				consolidated decision trees			
dataset	pruned	NRT	unpruned	PET	pruned	NRT	unpruned	PET
abalone19	0	0	0	0	0	**.4383**	**.4383**	.4352
yeast6	.566	.7335	.7335	.7339	.5342	.7203	.7937	**.7947**
yeast5	.8612	.8612	.872	.8726	.8648	.8648	**.9315**	.9284
yeast4	.4208	.4208	.4448	.2993	.3527	.5193	.7065	**.7072**
yeast2_vs_8	.1723	.1723	.0984	.0984	.7283	.7283	.7555	**.7597**
glass5	.8804	.8804	.8804	.8804	.9437	.9437	**.9774**	.9774
abalone9_vs_18	.3882	.3882	.5551	.5373	.0419	.6453	**.6934**	.6934
glass4	.5769	.5769	.5769	.5848	.742	.8184	.8508	**.8508**
ecoli4	.7777	.7777	.8159	.8201	.8321	.8321	.8369	**.8369**
glass2	.4427	.4427	.4427	.3464	.1171	**.624**	.6219	.6219
vowel0	**.9683**	**.9683**	.9627	.9627	.9286	.9286	.9367	.9367
page-blocks0	.9225	.9225	.923	.9177	.9167	.9167	.933	**.9331**
ecoli3	.6773	.6773	.7119	.696	.7472	.7472	**.8161**	.8133
yeast3	.8479	.8479	.8594	.8552	.8594	.8594	**.8801**	.8799
glass6	.794	.794	.7833	.7833	.8586	.8586	**.8625**	.8554
segment0	.9814	.9814	.9845	.9816	.9896	.9896	**.9933**	.9933
ecoli2	.8497	.8497	.8497	.8401	**.8764**	**.8764**	.876	.876
new-thyroid1	.946	.946	.946	.9155	.9571	.9571	.959	**.959**
new-thyroid2	.9327	.9327	.9327	.9327	**.9643**	**.9643**	.9587	.9593
ecoli1	.8538	.8538	.8538	.8481	.8623	.8623	.8707	**.8707**
vehicle0	**.9378**	**.9378**	.9334	.9282	.929	.929	.9266	.9257
glass0-1-2-3_vsvs_vs4-5-6	.7845	.7845	.7619	**.9274**	.8606	.8606	.8704	.8686
haberman	.3563	.3563	.3544	.3544	.4513	.5323	**.5917**	.5883
vehicle1	.6512	.6512	.625	.6137	.6537	.6537	**.6802**	.6779
vehicle2	.9453	.9453	.9405	.9413	**.9497**	**.9497**	.9472	.9469
vehicle3	.6728	.6728	.6839	.6722	.6628	.6628	**.6887**	.6876
yeast1	.6276	.6276	.6575	.6502	.6545	.6545	**.66**	.6581
glass0	.921	.921	**.9274**	.7568	.7714	.7714	.758	.7529
iris0	.9897	.9897	.9897	**.9897**	.9897	.9897	.9897	**.9897**
pima	.7023	.7023	**.7068**	.7026	.6849	.6849	.6802	.6774
ecoli0_vs_1	**.9831**	**.9831**	.976	.976	.9816	.9816	.9688	.9688
wisconsin	.9315	.9315	**.9449**	.8953	.926	.926	.9429	.8923
glass1	.7198	.7198	**.7211**	.7102	.706	.706	.6937	.6948
Mean	.7298	.7349	.7409	.728	.7375	.7999	**.8209**	.8185

Table G.6: Results for different pruning strategies for C4.5 over imbalanced datasets using GM as measure.

	base decision trees				consolidated decision trees			
dataset	pruned	NRT	unpruned	PET	pruned	NRT	unpruned	PET
abalone19	.0806	.1558	.1558	.1558	.4739	.4739	**.4791**	.4388
yeast6	.8042	.7262	.7253	.7253	**.8333**	**.8333**	.8119	.723
yeast5	.921	.9059	.8203	.8203	.9415	.9415	**.9466**	.8633
yeast4	.6952	.6236	.5629	.5635	**.8018**	**.8018**	.7994	.7269
yeast2_vs_8	.7313	.7039	.7039	.7039	**.7546**	**.7546**	.7418	.7493
glass5	.9274	.9316	.9316	.9316	.9507	.9507	.9518	**.9522**
abalone9_vs_18	.5534	.5456	.5456	.5456	**.6695**	**.6695**	.6655	.6423
glass4	**.8406**	.7966	.7993	.8045	.8268	.8268	.7241	.7224
ecoli4	.8184	**.8484**	.8221	.8221	.8009	.8009	.8295	.7636
glass2	**.66**	.6572	.6572	.6572	.6069	.6069	.584	.4832
vowel0	.9367	.9404	.9415	.9415	.9395	.9395	**.9427**	.8983
page-blocks0	**.9435**	.9244	.9206	.9195	.9364	.9364	.9367	.9171
ecoli3	.8338	.83	.7858	.7858	.8869	.8869	**.8898**	.8161
yeast3	.8783	.92	.909	.9087	.9238	.9238	**.9303**	.8941
glass6	**.8868**	.8071	.8048	.8071	.8367	.8367	.8795	.8132
segment0	.9891	.9898	**.9904**	.9893	.9831	.9831	.9862	.982
ecoli2	.8452	.8489	.8398	**.8489**	.8373	.8373	.8329	.814
new-thyroid1	**.9741**	.9595	.9595	.9595	.9556	.9556	.9574	.9315
new-thyroid2	.9473	.9916	.9916	**.9916**	.9496	.9496	.9502	.9416
ecoli1	**.9222**	.882	.8818	.8818	.8909	.8909	.8911	.8813
vehicle0	**.9294**	.9209	.9172	.9129	.9212	.9212	.9251	.9048
glass-0-1-2-3_vs_4-5-6	.8898	.7916	.7681	**.9135**	.908	.908	.8972	.8618
haberman	**.6189**	.5368	.5237	.5237	.5792	.5792	.5922	.5813
vehicle1	.7227	**.7316**	.7252	.7246	.7138	.7138	.7078	.6793
vehicle2	**.9649**	.9508	.9524	.9532	.9427	.9427	.9391	.9224
vehicle3	.7297	.7412	**.7445**	.7342	.7297	.7297	.7255	.6976
yeast1	.7012	**.7186**	.7165	.7165	.7014	.7014	.6991	.6965
glass0	.7617	**.9223**	.9092	.7872	.7731	.7731	.752	.738
iris0	.9897	.9897	.9897	**.9897**	.9897	.9897	.9877	.9877
pima	**.7174**	.693	.705	.6995	.7089	.7089	.7092	.7015
ecoli0_vs_1	.976	**.9831**	**.9831**	.9831	.9816	.9816	.9788	.9774
wisconsin	.9436	.9492	.9434	.9477	.9463	.9463	**.9546**	.9513
glass1	**.7002**	.6965	.69	.6632	.6922	.6922	.6931	.6827
Mean	.8132	.8065	.7975	.7974	**.8299**	**.8299**	.827	.7981

Table G.7: Results for different pruning strategies for CHAID* over SMOTE-preprocessed imbalanced datasets using GM as measure.

	base decision trees				consolidated decision trees			
dataset	pruned	NRT	unpruned	PET	pruned	NRT	unpruned	PET
abalone19	.1558	.1558	.1558	.1559	**.4794**	**.4794**	.4784	.4726
yeast6	.8048	.8048	.805	.805	**.8241**	**.8241**	.8204	.8171
yeast5	.9445	.9445	.9324	.9324	**.9608**	**.9608**	.9535	.9538
yeast4	.6521	.6521	.6519	.6527	**.7527**	**.7527**	.7425	.7413
yeast2_vs_8	.7402	.7402	.7403	.7403	.8077	.8077	.825	**.8257**
glass5	.872	.872	.872	.872	.9136	.9136	**.918**	.918
abalone9_vs_18	.5319	.5319	.5347	.5357	**.6157**	**.6157**	.6086	.5971
glass4	.8443	.8443	.8443	**.8443**	.8377	.8377	.8306	.834
ecoli4	.741	.741	.741	.741	**.8551**	**.8551**	.8469	.8477
glass2	.65	.65	.6535	.6535	**.734**	**.734**	.7055	.6976
vowel0	.9357	.9357	.9362	.9379	.9508	.9508	**.9513**	.9513
page-blocks0	.9428	.9428	.939	.9371	**.9442**	**.9442**	.939	.939
ecoli3	.7442	.7442	.761	.7625	**.8636**	**.8636**	.8566	.8484
yeast3	.8843	.8843	.8819	.8841	**.9067**	**.9067**	.8875	.8874
glass6	.8662	.8662	.8662	**.8691**	.8545	.8545	.8561	.8561
segment0	.9944	.9944	.9944	**.9944**	.9888	.9888	.9875	.9873
ecoli2	**.8809**	**.8809**	.8757	.8696	.8472	.8472	.8328	.8356
new-thyroid1	.9741	.9741	.9741	**.9741**	.9562	.9562	.9586	.9586
new-thyroid2	.9677	.9677	.9677	**.9677**	.9565	.9565	.9576	.9576
ecoli1	**.9125**	**.9125**	.9106	.9058	.8884	.8884	.8606	.8579
vehicle0	.9139	.9139	.9117	.9113	**.9296**	**.9296**	.9256	.9238
glass0-1-2-3_vs_4-5-6	.7726	.7726	.7726	.8721	**.8881**	**.8881**	.886	.8866
haberman	**.6074**	**.6074**	.5964	.5882	.5977	.5977	.5978	.59
vehicle1	.6838	.6838	.6803	.675	**.7229**	**.7229**	.699	.6942
vehicle2	.9486	.9486	**.9494**	.9446	.9473	.9473	.947	.9468
vehicle3	.7085	.7085	.6911	.6821	**.7174**	**.7174**	.7033	.7023
yeast1	.7135	.7135	**.7147**	.7125	.6928	.6928	.6669	.6638
glass0	.8668	.8668	**.8721**	.7644	.8032	.8032	.781	.7735
iris0	.9897	.9897	.9897	**.9897**	.9877	.9877	.9877	.9877
pima	**.7128**	**.7128**	.7096	.7093	.704	.704	.6785	.6776
ecoli0_vs_1	.9725	.9725	.9725	.9725	**.9779**	**.9779**	.9661	.9659
wisconsin	.9454	.9454	.9207	**.955**	.9435	.9435	.923	.9486
glass1	**.748**	**.748**	.7457	.7429	.7176	.7176	.7223	.7228
Mean	.8068	.8068	.805	.8047	**.8354**	**.8354**	.8273	.8263

Table G.8: Results for different pruning strategies for C45 over SMOTE-preprocessed imbalanced datasets using GM as measure.

APPENDIX G. FULL RESULT TABLES FOR THE THIRD EXPERIMENT FROM THE CONTRIBUTIONS TO THE CONSOLIDATION OF DECISION TREE ALGORITHMS

Appendix H

Performance of sixteen PART variants on UCI datasets

This appendix describes the experiment that was used to determine the best of the 16 PART variants proposed in Section 4.2. This work was published in [85].

This appendix contains the table with the characteristics for the 36 real world problem datasets used in the study mentioned in Section 4.2. The datasets whose names end with the subindex 2 represent two-class versions of multi-class datasets. This conversion was done by grouping all classes other than the one with the least examples (the same way it was done for imbalanced datasets in the reference work [53]). In these datasets the number between parentheses for the *#classes* column represents the number of classes on the original dataset.

The 36 real world problem datasets that were used for this experiments are described in Table H.1.

This experiment compares the 16 PART variants using five performance metrics, grouped into three groups: discriminating performance, structural complexity and computational cost.

For discriminating capacity the AUC and the error rate are used. For the structural complexity of the generated models two metrics are used: the number of internal nodes (number of decisions) of the ruleset, referred to as *Complexity*, and the average length of the rules (also measured in number of decisions), referred to as *Length*. The computational cost is measured as the time (in milliseconds) taken to build the classifier.

Only the analysis of statistical significance of the results is included.

For each of the metrics, a table with the average values obtained with each variant and a figure with the related CD diagram are given. These CD diagrams graphically show the presence or absence of significant differences based on the distance between the average ranks of two algorithms as [39] proposed.

		#features						
		#discrete						
dataset	#examples	#nominals	#ordinals	#continuous	#total	#classes	%min. class	size of min. class
breast-w	699		10		10	2	34.5	241
heartc	303	7	1	5	13	2	45.87	139
spam	4601			57	57	2	39.4	1813
hypo	3163	18		7	25	2	4.77	151
liver	345			6	6	2	42.03	145
lymph	148	17	1		18	4	1.35	2
lymph_2	148	17	1		18	2 (4)	41.22	61
credit_a	690	8		6	14	2	44.49	307
vehicle	846			18	18	4	23.52	199
vehicle_2	846			18	18	2 (4)	23.52	199
iris	150			4	4	3	33.33	50
iris_2	150			4	4	2 (3)	33.33	50
glass	214			9	9	7	4.2	9
glass_2	214			9	9	2 (7)	23.83	51
breast-y	286	5	4		9	2	29.72	85
voting	435	16			16	2	38.62	168
heart-h	294	8		5	13	2	36.05	106
hepatitis	155	13		6	19	2	20.65	32
credit_g	1000	10	3	7	20	2	30	300
soybean_15CL	290	34	1		35	15	3.45	10
soybean_15CL_2	290	34	1		35	2 (15)	13.79	40
segment2310	2310			19	19	7	14.29	330
segment2310_2	2310			19	19	2 (7)	14.29	330
segment210	210			19	19	7	14.29	30
segment210_2	210			19	19	2 (7)	14.29	30
sick-euthyroid	3164	18		7	25	2	9.26	293
bands	540	18		21	39	2	42.2	228
ks-vs-kp	3196	36			36	2	47.8	1528
optdigits_2	5620		64		64	2 (10)	9.9	556
car_2	1728		6		6	2 (4)	30	518
abalone_2	4177			8	8	2 (29)	8.6	359
solar_flare	1389	13			13	2	15.7	218
yeast_2	1484			8	8	2 (10)	28.9	429
splice_junction_2	3190	60			60	2 (3)	24.1	769
kddcup	4941	7		34	41	2	19.69	973
pima	768			8	8	2	34.9	268
Mean	1402.89				20.94	2.92	24.88	306.03
Median	694.5				18	2.00	23.97	199

Table H.1: UCI datasets for the analysis of sixteen PART variants.

	16 variants of PART
Nemenyi	3.8445
Bonferroni-Dunn	3.2936

Table H.2: Critical distance values for the Nemenyi and Bonferroni-Dunn tests used in this appendix.

The Friedman test (with the Iman-Davenport extension) [39] is used to discover whether or not significant differences appear in the behavior of the variants. When significant differences are found, the Nemenyi and the Bonferroni-Dunn classic tests are used, which are graphically represented in CD (Critical Difference) diagrams. The Nemenyi test is used to perform comparisons of all algorithms against each other ($n \times n$), whereas the Bonferroni-Dunn test is used to perform comparisons of one algorithm (the control algorithm) against the others ($1 \times n$ comparisons). The Bonferroni-Dunn test is applied twice for each metric using different control variants: the best ranking variant and the reference variant for the comparison, the original PART algorithm. The positive aspect of the Nemenyi and Bonferroni-Dunn tests is that their critical differences for significance are determined by the number of competitors, and thus, the critical differences are fixed and can easily be visually represented on CD diagrams. However, both of these tests are currently considered conservative. The results of these tests are complemented with more powerful tests (as proposed by García *et al.* in [65]), namely the Shaffer test to complement the Nemenyi test (both $n \times n$) and the Holm [64] test to complement the Bonferroni-Dunn test (both $1 \times n$). However, the results of these two tests cannot be represented on CD diagrams, so their results are discussed when referencing the CD diagrams. Holm's tests is only used once per measure, using the best ranking algorithm as control. The results of the two Bonferroni-Dunn tests are both represented by orange lines below the ranking line on CD diagrams, as their critical distance is the same, but a different tone for each line is used. The lighter line represents the results when the best ranking variant is used as control variant, and the darker line when PART is used as control.

For each of the Nemenyi and Bonferroni-Dunn tests, significance level, number of algorithms to be compared and number of datasets used, a critical value (maximum distance between the average ranks) is defined. For a significance level of 95% and 36 datasets, the critical values for the Nemenyi and Bonferroni-Dunn tests used in this appendix are shown in Table H.2. The critical values have been graphically represented in the CD diagrams to provide a visual reference. When the Nemenyi test is used, groups of variants that are not significantly different are connected by a line (above the x-axis). With respect to the Bonferroni-Dunn test, all variants with ranks outside the marked interval (under the x-axis) are significantly different from the control option.

H.1 Results for Discriminating Capacity Metrics

Table H.3 shows the average values for the two metrics used to assess the discriminating capacity for each of the 16 variants of the PART algorithm, whereas Figure H.1 shows two CD diagrams, one for the average ranks for each measure. Each AUC and Error value was calculated as an average of the values of the 36 datasets. When using a rank-based statistical analysis, the order in which the algorithms (the 16 different variants, in this case) appear based on their average values, and the order in which the algorithms appear based on their average ranks, are not necessarily the same. As an example, in Table H.3, the variant BF_AL_PR_DP (found in the lower row, last column) achieved the sixth best average value for AUC (86.02), but it obtained the best rank. This is because there may be some datasets that obtain much better AUC values for a particular variant, but that variant outperforms others in fewer datasets.

According to the AUC and Error results shown in Table H.3, the BF (Best-First) option appears to be a good choice for both measures as most variants using this option place in the top half. In fact, six out of eight variants using Best-First rank better than their Hill Climbing counterpart for AUC, and seven out of eight for Error. As expected, unpruned variants place better for the AUC measure. However, two variants using pruning (PR) placed in the top half, specifically variants combining pruning with developing trees further by not prioritizing pure nodes (DP). Surprisingly, pruning, a process designed to increase accuracy, does not appear to be relevant for the Error measure, as variants both pruning and not pruning place in top positions. With respect to the AUC measure, those variants that use pruning place in higher positions if combined with not prioritizing pure nodes, whereas pruning while prioritizing pure nodes does not appear to achieve good results.

For a more in-depth analysis of the differences in the results, an evaluation of statistically significant differences was carried out based on multiple tests. First, the Friedman test showed that significant differences existed between the 16 variants for both metrics, computing an adjusted p-value of 9.6473e-11 (test statistic value 17.14) and 0 (test statistic value 13.32) for AUC and Error, respectively. Therefore, the post hoc tests proposed by Demšar were applied, and obtained the CD diagrams shown in Figure H.1.

With respect to the AUC metric, the best ranking variant is BF_AL_PR_DP, which shows statistically significant differences compared to most Hill Climbing-based (HC) variants according to both post hoc tests. Most variants that use Best-First (BF) rank in the top half, except for those that combine pruning (PR) and pure nodes (PP). For this measure, there is a balance between variants that use the AL (All Leaves) option and those that use the TL (Treated Leaves) option, which are evenly divided in both halves of the ranking.

Results for the Error metric show another Best-First-based variant ranking first: BF_TL_PR_DP, showing significant differences with most variants using the HC option and two variants using BF, according to the post hoc tests.

	PART	HC_TL_NP_DP	HC_TL_PR_PP	HC_TL_PR_DP	HC_AL_NP_PP	HC_AL_NP_DP	HC_AL_PR_PP	HC_AL_PR_DP
AUC	87.34	87.08	80.08	80.16	81.68	81.68	76.84	76.7
Error	13.04	13.32	14.57	14.18	21.15	21.15	20.16	20.49
	BF_TL_NP_PP	BF_TL_NP_DP	BF_TL_PR_PP	BF_TL_PR_DP	BF_AL_NP_PP	BF_AL_NP_DP	BF_AL_PR_PP	BF_AL_PR_DP
AUC	87.22	87.15	80.58	86.01	83.03	87.25	77.79	86.02
Error	13.1	13.31	14.48	12.48	21	13.34	19.7	12.49

Table H.3: Average AUC and Error values for the 16 variants of the PART algorithm.

The variant ranking first for AUC also ranks second for Error, with both first and second ranking variants combining Best-First (BF), pruning (PR), and not prioritizing pure nodes (DP). For the Error metric, TL-based variants rank better than their AL counterpart.

The more powerful Shaffer and Holm post hoc tests confirm the pairwise statistically significant differences.

In summary, it should be noted that most variants that use Best-First rank better than their Hill-Climbing counterpart. In fact, variants combining Best-First, pruning and not prioritizing pure nodes rank first for both measures, showing significant differences with most Hill Climbing-based variants.

H.2 Results for Structural Complexity Metrics

This section analyzes the results obtained by the 16 variants of PART from the perspective of their explaining capacity, i.e., *Complexity* measured as the number of decisions throughout the whole classifier, and *Length* measured as the average number of conditions per branch or rule. Table H.4 includes the average values for both metrics. The values show that HC_AL_PR_DP and HC_AL_PR_PP, which are the two variants that combine the Hill Climbing algorithm as the Next Node to Develop criterion, the All Leaves value as the Leaf for Next Rule, and pruning criterion, produce the simplest models.

The CD diagrams presented in Figure H.2 show that these two variants (HC_AL_PR_DP and HC_AL_PR_PP) achieved significant differences when compared to most variants, including the original PART algorithm, according to the Nemenyi and both Bonferroni-Dunn tests. Statistically significant differences were found based on the Friedman test, which calculated a p-value of 1.4116e-10 and 1.8016e-10 for *Complexity* and *Length*, respectively. All post hoc tests, even the more powerful ones, agree on the pairwise significant differences.

Results suggest that in general, variants using HC are simpler than those using the BF option. Two variants using the BF option rank among the first group of variants that do not show significant differences among themselves. As expected, the top positions in both rankings are filled with variants that prune partial trees, as these trees will be equal or smaller than trees that are kept unpruned. In fact, all variants with the PR option rank better than their NP counterparts for both measures. In a similar fashion, for *Complexity*, with the

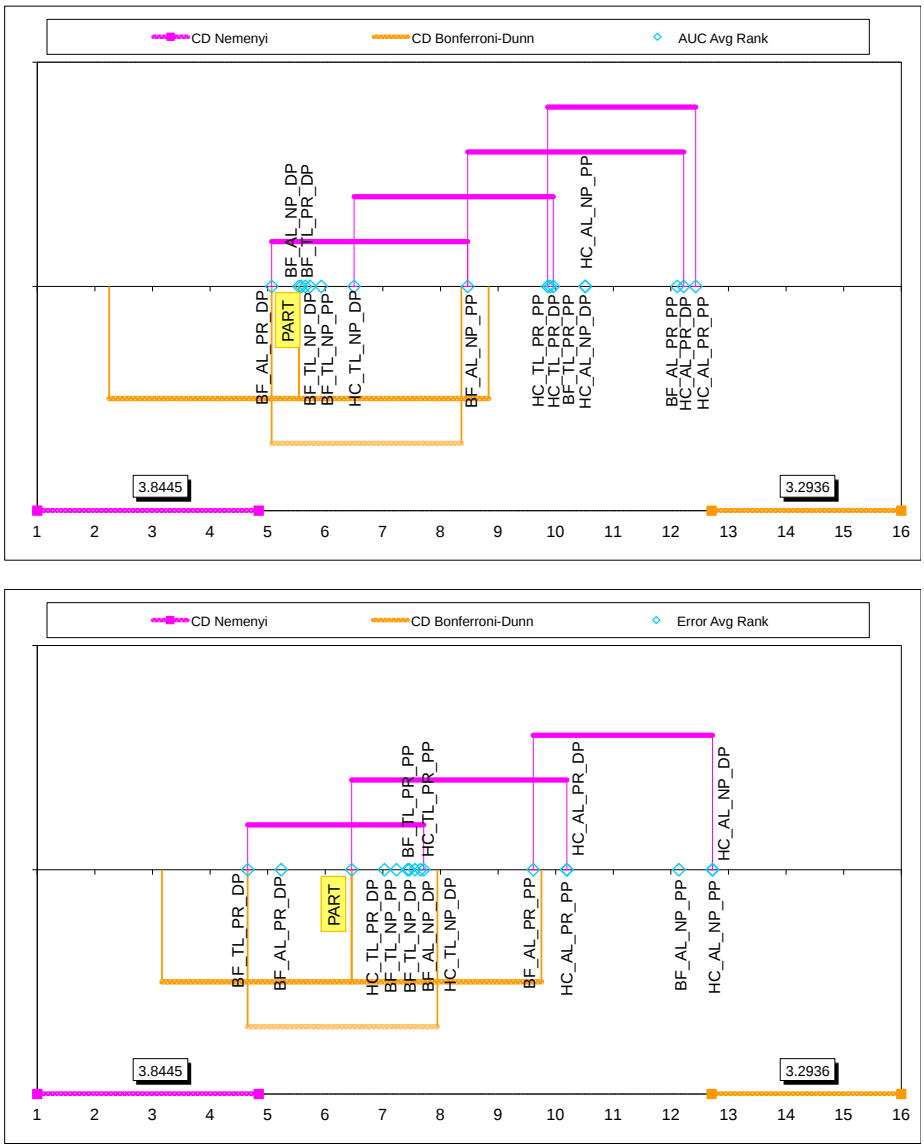

Figure H.1: CD diagrams for AUC and Error metrics comparing the 16 variants of the PART algorithm.

exception of one variant, all variants that use AL rank better than their TL counterparts. This occurs for every variant when *Length* is measured. This is expected however, as most of the time, AL should choose leaves at a level that is similar to or higher than TL, with the same or a smaller number of conditions, and it should therefore produce shorter rules than TL.

	PART	HC_TL_NP_DP	HC_TL_PR_PP	HC_TL_PR_DP	HC_AL_NP_PP	HC_AL_NP_DP	HC_AL_PR_PP	HC_AL_PR_DP
Complexity	25.01	28.79	6.29	6.87	7.57	7.57	3.85	3.83
Length	2.43	3.47	1.21	1.66	1.44	1.44	0.87	0.87
	BF_TL_NP_PP	BF_TL_NP_DP	BF_TL_PR_PP	BF_TL_PR_DP	BF_AL_NP_PP	BF_AL_NP_DP	BF_AL_PR_PP	BF_AL_PR_DP
Complexity	28.91	30.05	6.67	9.5	11.25	25.95	4.8	9.65
Length	2.95	3.88	1.38	2.31	2.12	3.18	1.17	2.02

Table H.4: Average *Complexity* and *Length* values for the 16 variants of the PART algorithm.

Finally, it should be noted that most variants ranking in the first group of variants, which show no significant differences for complexity measures, rank in very low positions for the discriminating capacity measures. For example, the two best-ranking variants for complexity measures achieve the two worst positions for both AUC and Error.

H.3 Results for Computational Cost Metrics

To evaluate the computational cost, the average time values in milliseconds for the 16 variants are provided in Table H.5. As seen in the table, the differences between some of the variants are of an order of magnitude. HC_AL_PR_DP and HC_AL_PR_PP (the variants achieving the simplest models according to the structural complexity metrics) were the fastest to build the ruleset. Unfortunately, these variants obtained very poor results for AUC and Error, being last for both metrics. It appears the short time taken to build the ruleset is not sufficient to obtain competitive classifiers.

In this case, the results of the Friedman test also indicated the presence of statistically significant differences, with a p-value of 1.2075e-10 (test statistic value 44.38). Figure H.3 shows the pairs between which the differences appeared. All variants belonging to the top ranking group that shows no significant differences use the AL option. This suggests that the variants using all leaves (and not just analyzed nodes) to create a rule cover more examples per rule, thus leading to the creation of fewer partial trees as the training example is covered by a smaller number of rules. The five worst-ranking variants (PART included) show one common trait, i.e., none of them prune the partial trees. This makes sense, as even though pruning trees requires extra computational effort, it results in leaf nodes that cover the same or a greater number of examples, and thus, the training sample is covered with fewer rules than when using unpruned partial trees. The Shaffer and Holm post hoc tests, which are more powerful, find another significant pairwise difference between BF_AL_PR_PP and BF_AL_PR_DP.

In summary, there are similarities between the structural complexity rankings and the time ranking. For example, the two variants creating the simplest classifiers are also the fastest to be built. However, as is the case with the structural complexity, the fastest variants to be built are also the worst classifying new examples. Variants pruning partial trees build more quickly than

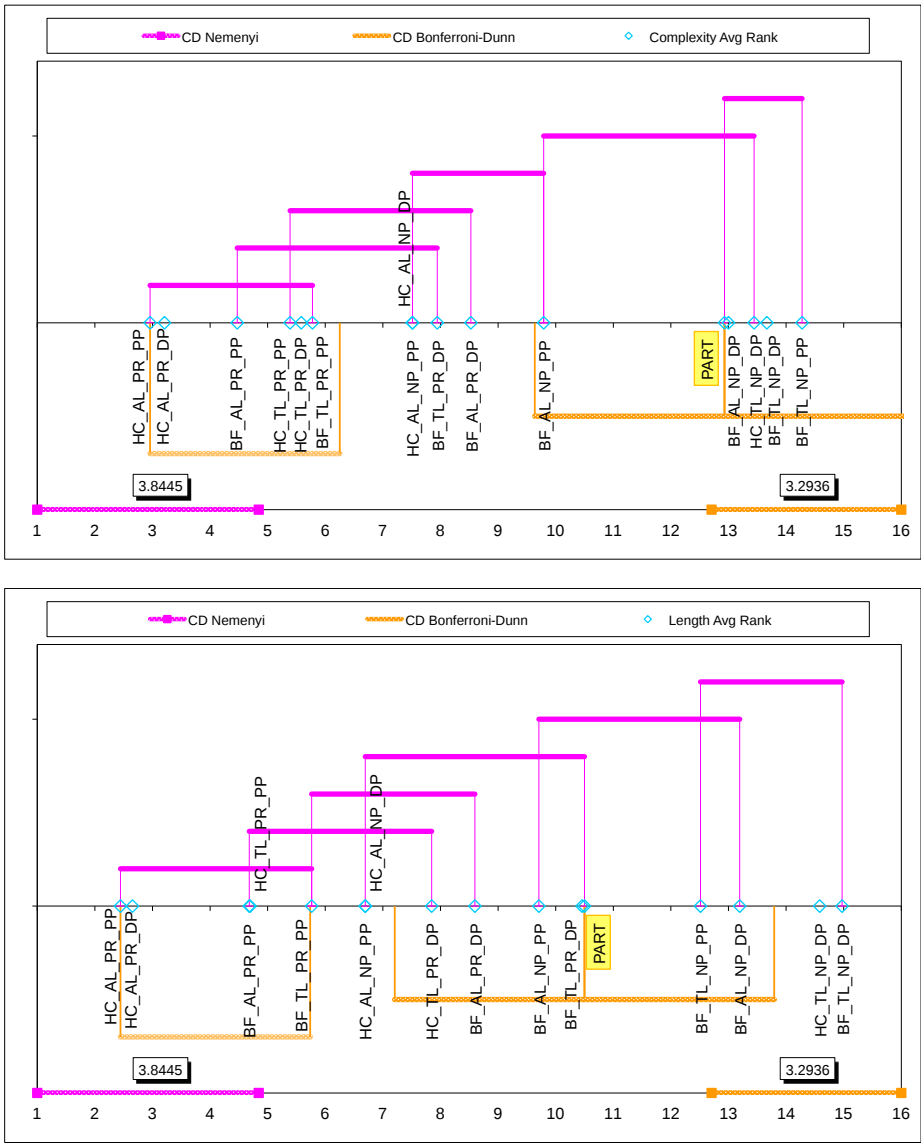

Figure H.2: CD diagrams for Complexity and *Length* metrics comparing the 16 variants of the PART algorithm.

their counterpart. At the same time, variants using the Hill Climbing option are faster than the same variant using Best-First.

	PART	HC_TL_NP_DP	HC_TL_PR_PP	HC_TL_PR_DP	HC_AL_NP_PP	HC_AL_NP_DP	HC_AL_PR_PP	HC_AL_PR_DP
Time	441.51	786.4	220.46	283.61	133.34	131.48	128.21	109.92
	BF_TL_NP_PP	BF_TL_NP_DP	BF_TL_PR_PP	BF_TL_PR_DP	BF_AL_NP_PP	BF_AL_NP_DP	BF_AL_PR_PP	BF_AL_PR_DP
Time	556.76	1436.84	233.67	536.71	207.52	759.68	176.76	435.74

Table H.5: Average Time values for the 16 variants of the PART algorithm.

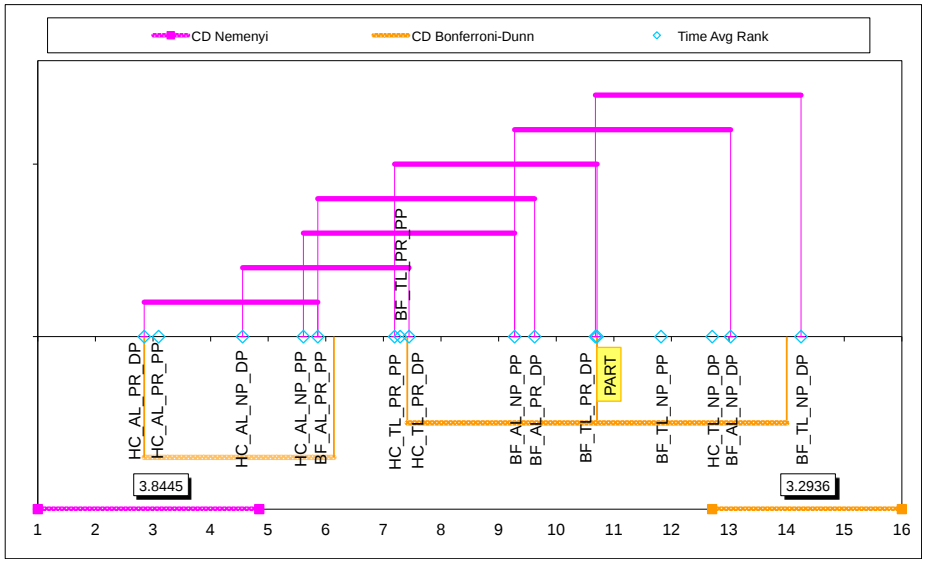

Figure H.3: CD diagram for Time metric comparing the 16 variants of the PART algorithm.

H.4 Selection and Naming of the Best Variant

This final section analyzes all of the variants of the PART algorithm from a global point of view, i.e., considering all of the metrics analyzed simultaneously, to select the best variant as a competitive alternative to the original PART algorithm. This is not an easy task because it deals with a multi-objective problem, where the best variant has to be selected according to all the metrics used.

As shown in previous subsections, the best variant is different depending on the point of view. The variants that produce the simplest models from the perspective of structural complexity are usually the fastest to be built, but they classify new examples significantly worse according to the AUC and Error measures than variants that are slower and more complex. This requires an approach to combine multiple measures in order to choose the best variant.

In order to find the best variant, the average "global" rank positions were computed, considering the average ranks for the five metrics. Table H.6 shows the average rank positions, and Figure H.4 shows these values graphically so that the distance between the different variants can be appreciated. In this

	PART	HC_TL_NP_DP	HC_TL_PR_PP	HC_TL_PR_DP	HC_AL_NP_PP	HC_AL_NP_DP	HC_AL_PR_PP	HC_AL_PR_DP
AUC	2	7	9	10	12.5	12.5	16	15
Error	3	10	8	4	15.5	15.5	12.5	12.5
Complexity	12	14	4	5	7.5	7.5	1	2
Length	12	15	4	8	6.5	6.5	1	2
Time	12	14	6	8	4	3	2	1
Avg ranks	8.2 (9)	12 (16)	6.2 (1)	7 (5)	9.2 (11)	9 (10)	6.5 (3.5)	6.5 (3.5)
	BF_TL_NP_PP	BF_TL_NP_DP	BF_TL_PR_PP	BF_TL_PR_DP	BF_AL_NP_PP	BF_AL_NP_DP	BF_AL_PR_PP	BF_AL_PR_DP
AUC	6	5	11	4	8	3	14	1
Error	5	6	7	1	14	9	11	2
Complexity	16	15	6	9	11	13	3	10
Length	13	16	5	11	10	14	3	9
Time	13	16	7	11	9	15	5	10
Avg ranks	10.6 (13)	11.6 (15)	7.2 (7)	7.2 (7)	10.4 (12)	10.8 (14)	7.2 (7)	6.4 (2)

Table H.6: Average global ranks (and rank positions) based on the analyzed five performance metrics for the 16 variants of the PART algorithm.

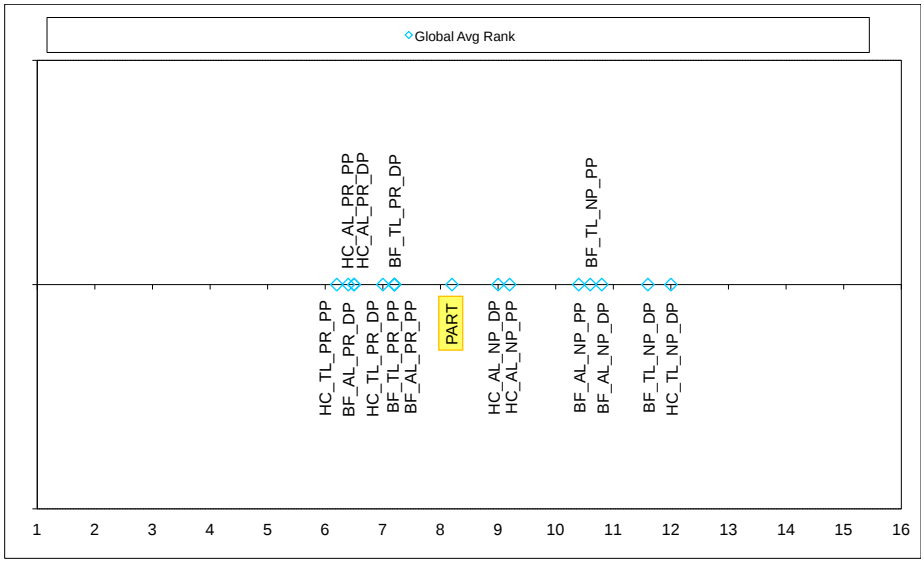

Figure H.4: Average global ranks for the 16 variants of the PART algorithm.

ranking, the HC_TL_PR_PP variant ranks best. However it should be noted that this is because it creates some of the simplest models at the cost of its ability of correctly classifying unseen instances. Its AUC and Error values are worse than most other variants. On the other hand, the second ranking variant, BF_AL_PR_DP, achieves the best rank for AUC and second best for Error, even if it creates complex models. Among the top five ranking variants, all but BF_AL_PR_DP are variants that create fast but less accurate models. For example, the third and fourth ranking variants also create very simple classifiers very fast, but then achieve the worst possible positions for AUC, and similar for Error. It can be argued that in a classifier algorithm the first priority should be

to create the best classifying models, and this should be prioritized more than other metrics. Therefore, the BF_AL_PR_DP variant was selected as the true alternative to the PART algorithm. In other words, its considered worth it to use the Best-First (BF) local search algorithm to determine the next node to develop in a partial tree, that all leaves (even undeveloped leaf-like nodes, the AL option) of the partial tree should be taken into account when selecting the most appropriate branch to extract the next rule, partial trees should be pruned (PR) and pure nodes should not be prioritized for analysis, and should be left for the end (DP). This variant is the complete opposite to PART with respect to the four proposed criteria. With the aim of having an appropriate name that is similar to the original PART name, this variant was named BFPART.

APPENDIX H. PERFORMANCE OF SIXTEEN PART VARIANTS ON UCI DATASETS

Appendix I

Full result tables for C4.5-based and CHAID*-based PART-like algorithms

This appendix includes the full tables of the results related to the C4.5-based and CHAID*-based algorithms (UnPART, BFPART, PART, and C4.5/CHAID*) for the six performance metrics used in the study: kappa, GM, AUC, *Number of Rules*, *Length*, and Time. For *Length* the unit of measurement is the average number of decisions per rule or tree branch. Computational cost is measured in milliseconds. Numbers in bold indicate the best value for that particular dataset and base algorithm (C4.5 or CHAID*).

Numbers in bold indicate the best average value for each dataset.

APPENDIX I. FULL RESULT TABLES FOR C4.5-BASED AND CHAID*-BASED PART-LIKE ALGORITHMS

dataset	C4.5-based				CHAID*-based			
	UnPART	BFPART	PART	C4.5	UnPART	BFPART	PART	CHAID*
nursery	.8424	.8482	**.8753**	.8341	.8478	.8477	**.8947**	.8684
abalone	**.1035**	.0918	.0683	.0962	.1314	.1326	.1497	**.1533**
ecoli	**.7359**	.6986	.7289	.703	.6981	**.7065**	.6814	.6805
lymphography	.5593	**.5758**	.4688	.5367	.5354	.5764	**.6374**	.5304
car	.9033	**.9068**	.8894	.7986	.8924	.89	.8971	**.9076**
zoo	**.9215**	**.9215**	**.9215**	**.9215**	.9218	.9218	.9218	**.9398**
flare	.6668	**.6751**	.647	.6676	.6707	.6559	.6613	**.6893**
glass	.5394	.5526	**.6076**	.5494	.5085	.5085	.5124	**.5391**
cleveland	**.2789**	.2187	.2444	.2257	**.2738**	.239	.2366	.2256
dermatology	**.9226**	**.9226**	**.9226**	.9045	.9335	.9335	.93	**.9383**
balance	.6447	.6447	**.6494**	.5922	.6175	.6209	**.6262**	.5996
penbased	.8818	**.8889**	.8737	.8838	.8331	.8311	.8355	**.8373**
newthyroid	**.8795**	**.8795**	.8525	.8519	.8618	.8618	.8618	**.8686**
hepatitis	.252	.252	**.5485**	.1115	.133	.0899	**.2285**	.1908
contraceptive	.2816	.2641	.2356	**.2845**	.2815	.2896	.2778	**.2909**
vehicle	.6421	.6248	**.6452**	.6185	.5789	.5707	**.5969**	.5902
haberman	.1186	.1186	.0924	**.1521**	.0265	**.0265**	**.0265**	.0265
wine	.8969	.8969	.8971	**.9222**	.8795	.8795	**.8966**	.8797
breast	**.2418**	**.2418**	.23	.233	**.297**	.297	.2663	**.297**
german	.3124	.3243	**.3333**	.3049	**.3158**	.315	.2705	.2714
iris	**.91**	**.91**	.9	.9	.92	**.92**	**.92**	.92
wisconsin	.8474	.8474	.8236	**.8515**	.8699	.8556	**.8907**	.841
tictactoe	.7803	.7573	**.843**	.677	.6817	.6789	**.7847**	.7605
pima	.3814	.3814	.4078	**.4175**	.4051	**.4051**	.3893	.4043
magic	.5186	**.519**	.4806	.5183	.5034	.4979	.4871	**.5266**
bupa	.2867	.2701	.2594	**.3124**	.2613	**.2613**	.2424	.2444
heart	**.5703**	.5384	.5239	.5636	.6225	**.6537**	.6537	.5489
australian	**.7156**	.7063	.6742	.6886	.6845	**.6851**	.6681	.6837
crx	**.7371**	.7184	.6651	.7194	**.7354**	.7194	.7116	.7172
ring	**.7702**	**.7702**	.7623	.7135	**.7482**	.742	.742	**.7482**
Mean	**.6047**	.5989	.6024	.5851	.589	.5871	**.5966**	.5906
Median	.6557	**.6599**	.6482	.6431	.6466	.6548	**.6575**	.64

Table I.1: Results for kappa over standard datasets.

dataset	C4.5-based				CHAID*-based			
	UnPART	BFPART	PART	C4.5	UnPART	BFPART	PART	CHAID*
abalone19	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**	**.0000**
yeast6	.5660	.5660	**.7333**	.5660	.1302	.1302	**.2359**	.1302
yeast5	.8846	.8742	**.8850**	.8612	.7967	.7957	**.8601**	.7715
yeast4	**.4731**	**.4731**	.4239	.4208	.3844	**.3844**	.2326	.3844
yeast2_vs_8	**.3137**	.1704	.1704	.1723	**.7274**	.7274	.7264	**.7274**
glass5	.8787	.8787	.8787	**.8804**	**.9876**	.9876	.9876	**.9876**
abalone9_vs_18	**.4173**	.3897	.4157	.3882	**.2983**	.2983	.2930	.2975
glass4	**.5836**	**.5836**	**.5836**	.5769	.3397	**.3397**	**.3397**	.3030
ecoli4	**.7791**	**.7791**	.7786	.7777	.7821	.7821	**.8405**	.8123
glass2	**.4597**	**.4597**	**.4597**	.4427	**.0000**	**.0000**	**.0000**	**.0000**
vowel0	.9149	.9149	.9207	**.9683**	.8552	.9112	**.9194**	.8722
page-blocks0	.9136	.8944	.8932	**.9225**	.8879	.9001	**.9096**	.9002
ecoli3	.7227	.7227	**.7290**	.6773	.7003	**.7003**	**.7003**	.6749
yeast3	**.8567**	**.8567**	.8097	.8479	.8537	.8537	.8552	**.8670**
glass6	**.7940**	**.7940**	.7874	**.7940**	.7946	**.7946**	**.7946**	.7920
segment0	.9858	.9858	**.9893**	.9814	.9794	.9796	**.9867**	.9829
ecoli2	.8041	.8041	.7963	**.8497**	**.7564**	.7564	.7564	**.7564**
new-thyroid1	.9394	.9394	.9394	**.9460**	.9529	**.9529**	**.9529**	.9503
new-thyroid2	.9206	.9206	.9206	**.9327**	.9328	.9328	.9328	**.9355**
ecoli1	**.8590**	**.8590**	.8411	.8538	.8062	.8062	.8062	**.8221**
vehicle0	.9029	.9242	.9267	**.9378**	.9209	**.9213**	.8902	.9023
glass-0-1-2-3_vs_4-5-6	.8711	.8711	.8682	**.9210**	.8638	.8660	.8576	**.8766**
haberman	.3252	.3252	.2876	.3563	.0931	.0931	.0931	.0931
vehicle1	.6362	.6112	.3773	.6512	.5063	.4975	.5113	.5033
vehicle2	.9421	.9514	.9499	.9453	.9234	.9284	.9394	.9221
vehicle3	.6581	.6489	.5071	**.6728**	.5250	.4651	.4939	**.5542**
yeast1	.6093	.6093	.5333	**.6276**	.6048	**.6068**	.5996	.5889
glass0	.7703	.7742	**.7967**	.7845	.7726	.7726	**.7752**	.7726
iris0	**.9897**	**.9897**	**.9897**	**.9897**	.9897	**.9897**	**.9897**	.9897
pima	.6724	.6795	.6973	**.7023**	.6879	.6879	.6709	**.6953**
ecoli0_vs_1	**.9831**	**.9831**	.9723	**.9831**	**.9831**	.9831	.9831	**.9831**
wisconsin	**.9365**	.9345	.9347	.9315	.9445	**.9561**	.9521	.9411
glass1	.7065	.6974	.6503	**.7198**	.6493	**.6732**	**.6732**	.6278
Mean	.7294	.7232	.7105	**.7298**	.6797	.6810	**.6836**	.6793
Median	.7940	.7940	**.7963**	.7940	.7821	.7821	**.7946**	.7726

Table I.2: Results for GM over imbalanced datasets.

dataset	C4.5-based				CHAID*-based			
	UnPART	BFPART	PART	C4.5	UnPART	BFPART	PART	CHAID*
abalone19	.0806	.0811	**.1560**	.1558	.0809	.1624	**.1625**	.1600
yeast6	.8042	.8042	.7855	**.8048**	**.6794**	.6122	.6283	.5637
yeast5	.9210	.9210	.9198	**.9445**	.8428	**.8571**	.7098	.7987
yeast4	.6952	.6782	**.6954**	.6521	.6151	**.6814**	.6043	.5417
yeast2_vs_8	.7313	.7323	.7171	**.7402**	**.7047**	.6653	.6607	.6509
glass5	**.9274**	**.9274**	.9249	.8720	.9242	.6687	.8688	**.9266**
abalone9_vs_18	.5534	**.5845**	.5773	.5319	.6157	.6260	**.6714**	.5169
glass4	.8406	.8406	.8406	**.8443**	.6714	**.7097**	**.7097**	.6673
ecoli4	.8184	.8184	**.8468**	.7410	.8123	.7712	**.9060**	.8142
glass2	.6600	.6600	**.6754**	.6500	.6053	.4813	.4993	**.6873**
vowel0	**.9367**	.9311	.8956	.9357	**.8831**	.8639	.8343	.8452
page-blocks0	.9435	**.9439**	.9438	.9428	.9291	**.9374**	.9317	.9174
ecoli3	.8338	.8338	**.8387**	.7442	.8108	.7867	.8200	**.8365**
yeast3	.8783	.8790	**.8939**	.8843	.8994	.8867	**.9034**	.8996
glass6	**.8868**	**.8868**	.8255	.8662	**.8596**	.8596	.8454	.8019
segment0	.9891	.9891	.9878	**.9944**	.9919	**.9924**	.9880	.9921
ecoli2	.8452	.8577	.8797	**.8809**	.8599	**.9024**	.8949	.8785
new-thyroid1	**.9741**	**.9741**	.9713	**.9741**	.9624	.9624	**.9651**	.9624
new-thyroid2	.9473	.9445	.9449	**.9677**	.9888	.9888	.9647	**.9916**
ecoli1	**.9222**	.9031	.8923	.9125	.8837	.8837	.8838	**.9111**
vehicle0	.9294	**.9302**	.9247	.9139	**.9243**	.9163	.9102	.9018
glass-0-1-2-3_vs_4-5-6	**.8898**	**.8898**	.8688	.8668	**.9085**	.8825	.8887	.8922
haberman	.6189	.6085	.6381	.6074	.5368	.5368	.5368	.5368
vehicle1	.7227	.7033	.7367	.6838	.7526	.7501	.7239	.7156
vehicle2	.9649	.9649	.9695	.9486	.9508	.9464	.9262	.9398
vehicle3	**.7297**	.7292	.7183	.7085	.7454	**.7542**	.7514	.7477
yeast1	.7012	.6972	.6881	**.7135**	.6988	.6983	.7086	**.7199**
glass0	.7617	.7617	.7617	**.7726**	.7913	.7913	.7906	**.7916**
iris0	**.9897**	**.9897**	**.9897**	**.9897**	.9897	**.9897**	**.9897**	.9897
pima	**.7174**	.7145	.6840	.7128	.7025	.7028	**.7268**	.6980
ecoli0_vs_1	**.9760**	**.9760**	**.9760**	.9725	**.9831**	.9831	.9831	**.9831**
wisconsin	.9436	.9436	.9434	**.9454**	.9362	**.9662**	.9552	.9393
glass1	.7002	.7002	.7158	**.7480**	.6885	.6885	.6811	**.6902**
Mean	**.8132**	.8121	.8129	.8068	**.7948**	.7850	.7886	.7851
Median	.8452	.8577	.8468	**.8662**	**.8428**	.7913	.8343	.8142

Table I.3: Results for GM over imbalanced datasets preprocessed with SMOTE.

dataset	C4.5-based				CHAID*-based			
	UnPART	BFPART	PART	C4.5	UnPART	BFPART	PART	CHAID*
nursery	.9678	.9709	**.9723**	.961	.9736	.9741	**.9828**	.9781
abalone	.582	**.5878**	.5525	.5668	.6824	.6823	**.6878**	.6847
ecoli	**.9048**	.8818	.898	.878	.9099	**.9132**	.9036	.9125
lymphography	**.8402**	.8192	.7544	.8193	.8369	.8363	**.8459**	.8163
car	**.987**	.987	.9858	.9681	.9843	.9844	.9898	**.9914**
zoo	.9804	.9804	.9804	**.9814**	.986	.986	.986	**.9869**
flare	.9177	**.9243**	.9096	.9154	.9225	.9212	.9213	**.9251**
glass	.7832	.7922	.8201	**.823**	.8009	**.8009**	.7992	.7777
cleveland	**.7082**	.6549	.6695	.6604	.7346	.7225	**.7614**	.7161
dermatology	**.9667**	**.9667**	**.9667**	.9572	**.9839**	.9762	.9751	.9707
balance	.8616	.8616	**.8953**	.8359	.8466	**.8606**	.8604	.8328
penbased	.9448	.9487	.9458	**.9523**	.9476	.9426	**.9481**	.9436
newthyroid	.9318	.9318	.93	**.9331**	.9177	.9177	.9177	**.929**
hepatitis	.686	.686	**.7744**	.469	.6001	.5858	**.6735**	.6103
contraceptive	**.6896**	.6894	.6485	.6683	.6896	.6878	**.71**	.6935
vehicle	**.8833**	.8751	.8789	.8462	.8752	.8704	**.8798**	.8761
haberman	.57	.57	**.5883**	.5714	.5117	.5117	**.539**	.5171
wine	.9469	.9469	.9522	**.9609**	.9473	**.9473**	.9442	.9381
breast	**.6299**	**.6299**	.6096	.6016	.6709	.6709	.6487	**.6712**
german	.7028	**.7149**	.6961	.6643	.7094	.7091	**.7179**	.6815
iris	.964	.964	**.9687**	.9538	**.9683**	.9683	.9683	.9577
wisconsin	.954	.954	**.9587**	.9323	.9381	.9614	**.9623**	.937
tictactoe	.9397	.9313	**.9722**	.9002	.8952	.8991	**.9352**	.9275
pima	.7574	.7574	**.7688**	.7529	.7444	.7501	**.7751**	.755
magic	**.8165**	.8149	.8113	.7655	**.8003**	.7974	.7956	.7999
bupa	.6709	.6418	.6536	**.6897**	.6502	.6502	.6345	**.6567**
heart	.7651	.7706	.7894	**.8021**	.8166	.8381	**.8395**	.8019
australian	**.8762**	.8548	.84	.8578	.8646	.8925	**.9033**	.8745
crx	**.9059**	.8964	.8772	.8664	.8739	.8821	**.8969**	.8781
ring	**.899**	**.899**	.8739	.8489	.8742	.8714	.8714	**.8776**
Mean	**.8345**	.8301	.8314	.8134	.8319	.8337	**.8425**	.8306
Median	**.8797**	.8684	.8755	.8475	.8693	.8709	**.8756**	.8753

Table I.4: Results for AUC over standard datasets.

dataset	C4.5-based				CHAID*-based			
	UnPART	BFPART	PART	C4.5	UnPART	BFPART	PART	CHAID*
abalone19	**.5000**	**.5000**	**.5000**	**.5000**	.5000	.5000	**.5001**	.5000
yeast6	.7846	.7846	**.8542**	.7855	.5798	.5798	**.6040**	.5798
yeast5	.9692	**.9694**	.9469	.9584	.8248	.8237	**.8782**	.8242
yeast4	.7583	.7583	**.8651**	.7799	.7381	.7381	**.8107**	.7381
yeast2_vs_8	**.6239**	.5818	.6057	.5718	**.7728**	.7728	.7704	.7720
glass5	.9927	.9927	.9927	**.9951**	**.9878**	.9878	.9878	**.9878**
abalone9_vs_18	.6831	**.7312**	.7218	.6547	.6384	.6384	**.6847**	.6371
glass4	.6517	.6517	.6517	**.6925**	.6046	**.6046**	**.6046**	.5988
ecoli4	.7659	.7659	**.8401**	.7643	.8187	.8187	**.8866**	.8655
glass2	.7724	.7724	**.7780**	.7535	**.4975**	**.4975**	**.4975**	**.4975**
vowel0	.9449	.9405	.9393	**.9706**	.8970	**.9464**	.9320	.8865
page-blocks0	**.9765**	.9764	.9604	.9595	.9624	.9647	**.9654**	.9552
ecoli3	.7726	.7726	**.8412**	.7638	.9071	**.9071**	**.9071**	.9014
yeast3	.9132	.9132	**.9388**	.9054	.9144	**.9144**	.9081	.9088
glass6	.7914	.7914	**.8171**	.7914	**.7895**	.7895	.7895	.7873
segment0	.9860	.9860	**.9901**	.9844	.9774	.9776	**.9904**	.9832
ecoli2	.8296	.8296	.8296	**.8459**	**.8288**	.8288	.7824	.7966
new-thyroid1	.9548	.9548	.9548	**.9655**	.9544	**.9544**	**.9544**	.9516
new-thyroid2	.9349	.9349	.9349	**.9472**	.9345	.9345	.9345	**.9373**
ecoli1	.9131	.9159	**.9347**	.8807	.9356	**.9356**	**.9356**	.9223
vehicle0	.9297	**.9544**	.9477	.9468	.9460	.9483	.9352	**.9490**
glass-0-1-2-3_vs_4-5-6	.9005	.9005	.8894	**.9209**	.9564	.9564	**.9583**	.9284
haberman	.5700	.5700	.5883	.5714	.5117	.5117	.5390	.5117
vehicle1	.6864	.7123	.7383	.7099	.6718	.6786	.7085	.6751
vehicle2	.9448	.9489	.9448	.9398	.9632	.9584	.9530	.9564
vehicle3	.7628	.7666	**.8012**	.7357	.7455	.7516	**.7857**	.7788
yeast1	.7083	.7134	**.7425**	.7017	.6768	.6746	**.7036**	.7007
glass0	.7780	.7834	**.8231**	.8119	.7992	.7992	.7789	**.8004**
iris0	**.9900**	**.9900**	**.9900**	**.9900**	**.9900**	**.9900**	**.9900**	**.9900**
pima	.7604	.7574	**.7688**	.7529	.7462	.7501	**.7751**	.7550
ecoli0_vs_1	**.9832**	**.9832**	.9726	**.9832**	**.9832**	**.9832**	**.9832**	**.9832**
wisconsin	.9647	.9651	.9666	**.9700**	.9593	.9606	.9571	**.9632**
glass1	.7112	.7170	.6984	**.7192**	**.6977**	.6831	.6831	.6882
Mean	.8245	.8268	**.8415**	.8250	.8094	.8109	**.8205**	.8094
Median	.7914	.7914	**.8542**	.8119	.8248	.8237	**.8782**	.8242

Table I.5: Results for AUC over imbalanced datasets.

dataset	C4.5-based				CHAID*-based			
	UnPART	BFPART	PART	C4.5	UnPART	BFPART	PART	CHAID*
abalone19	.5640	**.6000**	.5299	.5363	.5778	**.6585**	.5965	.6054
yeast6	**.8677**	**.8677**	.8561	.8490	**.8023**	.7879	.8001	.7078
yeast5	.9633	.9633	.9312	**.9646**	**.9043**	.8819	.7965	.8122
yeast4	**.8557**	.8426	.8494	.8107	**.8293**	.8253	.7932	.7136
yeast2_vs_8	.7756	.7761	**.8423**	.7259	.7326	.7249	**.7454**	.6726
glass5	**.9415**	**.9415**	**.9415**	.8890	.9220	.8646	.8829	**.9402**
abalone9_vs_18	.7274	**.7436**	.6611	.6579	.6308	.6973	**.7252**	.5543
glass4	.8583	.8583	.8583	**.8621**	.7173	**.7490**	**.7490**	.6848
ecoli4	.8485	.8485	**.8874**	.7783	.8425	.8270	**.9111**	.8021
glass2	.7360	.7360	**.7703**	.7325	.6801	.6288	.6349	**.7195**
vowel0	.9395	.9439	.9046	**.9523**	**.8859**	.8751	.8830	.8582
page-blocks0	.9727	**.9759**	.9697	.9582	.9678	**.9731**	.9682	.9549
ecoli3	**.8929**	**.8929**	.8793	.7989	**.8874**	.8696	.8722	.8635
yeast3	.9345	**.9348**	.9340	.9158	.9387	.9277	**.9478**	.9290
glass6	.8323	.8323	.8073	**.8908**	.8386	.8386	**.8759**	.7245
segment0	.9883	.9883	.9895	**.9941**	.9917	.9916	.9892	**.9954**
ecoli2	.8906	.8834	**.8957**	.8768	.8447	.8855	**.8979**	.8909
new-thyroid1	**.9802**	**.9802**	.9718	**.9802**	.9603	.9603	**.9659**	.9615
new-thyroid2	.9516	.9536	.9460	**.9714**	.9865	.9865	.9794	**.9921**
ecoli1	.9235	.9028	.9133	**.9261**	.9160	.9129	.9258	**.9282**
vehicle0	**.9459**	.9448	.9355	.9320	**.9469**	.9329	.9363	.9303
glass-0-1-2-3_vs_4-5-6	**.8971**	**.8971**	.8714	.8953	**.9261**	.8766	.9027	.9223
haberman	.6243	.6510	.6518	.6262	.6031	.6031	.5992	.6081
vehicle1	.7638	.7335	.7814	.6967	.7836	.7826	.7650	.7536
vehicle2	.9623	.9666	.9708	.9600	.9608	.9524	.9556	.9525
vehicle3	.7807	**.7858**	.7708	.7378	.7720	.7820	**.7924**	.7837
yeast1	.7507	**.7550**	.7523	.7362	.7160	.7210	.7487	**.7607**
glass0	.7673	.7686	.7686	**.7716**	.7802	.7802	.7858	**.8134**
iris0	**.9900**	**.9900**	**.9900**	**.9900**	**.9900**	**.9900**	**.9900**	**.9900**
pima	**.7686**	.7618	.7511	.7374	.7431	.7464	**.7760**	.7514
ecoli0_vs_1	**.9762**	**.9762**	**.9762**	.9728	**.9832**	**.9832**	**.9832**	**.9832**
wisconsin	.9725	.9725	**.9738**	.9706	.9527	**.9760**	.9682	.9488
glass1	.7175	.7175	.7490	**.7717**	.7005	.7005	.7058	**.7170**
Mean	.8594	**.8602**	.8570	.8445	.8398	.8392	**.8439**	.8250
Median	**.8906**	.8834	.8793	.8768	.8447	.8646	**.8759**	.8134

Table I.6: Results for AUC over imbalanced datasets preprocessed with SMOTE.

dataset	C4.5-based				CHAID*-based			
	UnPART	BFPART	PART	C4.5	UnPART	BFPART	PART	CHAID*
nursery	**26.60**	28.40	50.20	65.40	**18.00**	18.20	21.80	37.60
abalone	**46.60**	46.80	80.80	101.00	4.84	4.80	**4.40**	11.12
ecoli	12.20	**10.20**	14.20	19.40	**6.60**	6.80	6.80	13.20
lymphography	**6.20**	6.80	9.40	14.40	**5.48**	6.80	7.00	11.20
car	**38.20**	39.00	53.00	114.20	**23.80**	24.00	28.00	48.20
zoo	**7.20**	**7.20**	**7.20**	8.40	8.40	8.40	**8.20**	10.40
flare	8.80	**8.40**	33.40	45.80	**7.24**	12.00	15.40	7.80
glass	12.20	**12.00**	16.00	22.20	**5.80**	**5.80**	**5.80**	9.96
cleveland	19.24	**19.20**	34.40	37.00	**3.80**	4.00	5.20	9.20
dermatology	**7.60**	**7.60**	**7.60**	23.40	**8.04**	9.40	9.20	15.20
balance	**12.40**	**12.40**	32.60	37.00	7.00	**6.20**	7.20	14.40
penbased	23.80	**23.60**	29.20	50.00	22.60	**22.40**	23.80	61.12
newthyroid	**4.40**	**4.40**	**4.40**	6.80	**5.00**	**5.00**	**5.00**	7.80
hepatitis	**3.40**	**3.40**	4.20	4.80	**2.08**	**2.08**	2.28	4.16
contraceptive	33.60	**28.60**	143.00	111.00	6.00	**5.60**	9.40	15.60
vehicle	**20.60**	22.40	31.40	62.00	11.44	**11.40**	14.20	37.36
haberman	**2.60**	**2.60**	2.80	2.80	**1.40**	**1.40**	**1.40**	1.44
wine	**4.20**	**4.20**	**4.20**	5.20	4.20	4.20	**4.00**	7.80
breast	**3.80**	**3.80**	14.60	13.40	3.80	3.80	**3.20**	4.00
german	18.20	**15.40**	60.20	65.00	**4.80**	5.40	6.76	14.40
iris	**3.20**	**3.20**	4.40	4.20	3.20	**3.00**	**3.00**	4.20
wisconsin	**5.20**	**5.20**	8.60	31.60	4.80	5.20	**4.20**	6.32
tictactoe	**21.00**	22.80	39.80	77.40	8.80	**8.40**	12.80	31.84
pima	**6.40**	**6.40**	7.40	21.20	**4.20**	4.40	4.40	7.80
magic	8.00	**7.80**	10.00	39.00	**4.00**	4.24	5.68	9.00
bupa	8.20	8.40	**6.80**	24.60	3.40	3.40	**2.80**	4.20
heart	**7.20**	8.00	16.80	21.40	**4.08**	4.24	5.20	8.60
australian	11.40	**11.20**	26.80	20.40	**4.20**	5.00	5.40	4.96
crx	**7.20**	**7.20**	31.60	19.20	**3.64**	4.24	5.60	5.08
ring	**7.80**	**7.80**	11.40	25.00	**2.00**	**2.00**	**2.00**	15.04
Mean	13.25	**13.15**	26.55	36.44	**6.75**	7.06	8.00	14.63
Median	**8.10**	8.20	15.30	24.00	**4.82**	5.10	5.64	9.58

Table I.7: Results for *Number of Rules* over standard datasets.

dataset	C4.5-based				CHAID*-based			
	UnPART	BFPART	PART	C4.5	UnPART	BFPART	PART	CHAID*
abalone19	**1**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	1.60	**1.00**
yeast6	**4.00**	**4.00**	4.80	5.20	**1.60**	**1.60**	2.00	**1.60**
yeast5	**7.20**	7.40	7.40	9.20	**4.00**	**4.00**	**4.00**	5.00
yeast4	**5.00**	**5.00**	6.80	8.40	**3.40**	**3.40**	3.60	3.80
yeast2_vs_8	**1.40**	2.00	2.40	1.80	**2.00**	**2.00**	2.20	2.80
glass5	**4.00**	**4.00**	**4.00**	**4.00**	**3.00**	**3.00**	**3.00**	**3.00**
abalone9_vs_18	**4.40**	4.80	8.00	7.40	**2.40**	**2.40**	2.80	2.80
glass4	**4.00**	**4.00**	**4.00**	5.60	**2.60**	**2.60**	**2.60**	3.20
ecoli4	**3.00**	**3.00**	3.80	3.60	**3.00**	**3.00**	3.20	4.20
glass2	**5.40**	**5.40**	5.60	7.00	**1.20**	**1.20**	**1.20**	**1.20**
vowel0	**5.20**	5.60	6.20	7.80	**4.40**	4.52	5.08	12.68
page-blocks0	11.60	**11.00**	17.20	36.20	**7.16**	8.60	10.40	33.64
ecoli3	4.40	4.40	**4.00**	6.20	**3.00**	**3.00**	**3.00**	3.20
yeast3	**5.60**	**5.60**	9.00	10.80	**3.20**	**3.20**	3.40	4.00
glass6	**3.00**	**3.00**	3.40	3.40	**3.00**	**3.00**	**3.00**	3.60
segment0	**5.80**	**5.80**	6.40	9.60	**5.20**	5.40	6.80	8.20
ecoli2	**4.40**	**4.40**	4.80	7.40	3.20	3.20	**2.00**	4.80
new-thyroid1	**3.00**	**3.00**	**3.00**	5.40	**3.80**	**3.80**	**3.80**	4.80
new-thyroid2	**3.40**	**3.40**	**3.40**	4.40	**3.80**	**3.80**	**3.80**	4.60
ecoli1	**4.80**	5.00	5.20	7.00	**3.00**	**3.00**	**3.00**	4.80
vehicle0	**10.20**	**10.20**	12.20	20.00	**7.20**	7.40	8.44	17.08
glass-0-1-2-3_vs_4-5-6	**4.20**	**4.20**	4.60	5.80	**3.00**	**3.00**	**3.00**	5.00
haberman	**2.60**	**2.60**	2.80	2.80	**1.40**	**1.40**	**1.40**	**1.40**
vehicle1	**11.20**	12.80	11.80	35.60	**3.80**	4.00	4.80	5.40
vehicle2	**7.40**	9.00	9.00	15.80	**6.40**	7.00	7.80	15.20
vehicle3	10.40	**9.60**	11.20	41.80	**4.60**	**4.60**	6.80	8.40
yeast1	9.00	9.60	**8.80**	21.40	3.88	**3.80**	5.20	6.00
glass0	5.60	**5.20**	7.20	9.80	2.64	2.64	**2.08**	3.64
iris0	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**
pima	**6.00**	6.40	7.40	21.20	**4.20**	4.40	4.40	7.80
ecoli0_vs_1	**2.00**	**2.00**	2.80	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**
wisconsin	**7.00**	7.40	9.00	31.60	5.60	6.20	**5.00**	11.40
glass1	5.60	**5.20**	6.00	10.80	3.00	**2.60**	**2.60**	5.00
Mean	**5.27**	5.39	6.22	11.27	**3.44**	3.54	3.82	6.16
Median	**4.80**	5.00	5.60	7.40	**3.00**	**3.00**	**3.00**	4.60

Table I.8: Results for *Number of Rules* over imbalanced datasets.

dataset	C4.5-based				CHAID*-based			
	UnPART	BFPART	PART	C4.5	UnPART	BFPART	PART	CHAID*
abalone19	**23**	24.00	34.60	67.60	**18.80**	19.80	20.60	74.80
yeast6	**10.20**	**10.20**	10.60	21.20	**10.20**	11.00	12.00	36.00
yeast5	**6.80**	**6.80**	7.60	11.40	**9.40**	9.60	12.60	11.60
yeast4	15.40	**14.80**	18.20	41.00	**13.80**	14.20	14.80	45.40
yeast2_vs_8	**7.40**	**7.40**	8.00	12.80	**7.40**	7.60	9.00	14.80
glass5	**3.20**	**3.20**	4.20	6.20	**3.80**	4.20	**3.80**	8.80
abalone9_vs_18	20.60	**19.40**	26.00	49.00	**11.00**	11.36	12.20	29.40
glass4	**4.80**	**4.80**	**4.80**	8.20	**3.40**	**3.40**	**3.40**	8.00
ecoli4	**5.00**	**5.00**	5.60	9.80	**4.20**	**4.20**	5.40	9.40
glass2	**7.60**	**7.60**	8.00	15.60	**3.60**	3.80	4.80	9.80
vowel0	6.60	6.80	**6.40**	12.20	**8.76**	9.44	10.80	13.00
page-blocks0	29.80	**28.40**	42.00	116.80	**19.20**	20.80	33.60	88.40
ecoli3	**7.20**	**7.20**	8.00	12.80	4.20	4.40	**4.00**	9.00
yeast3	**13.00**	13.20	15.20	31.60	**7.20**	8.00	8.20	23.92
glass6	4.60	4.60	**4.00**	8.80	**3.60**	**3.60**	4.00	6.60
segment0	**7.20**	7.80	7.60	13.80	**6.00**	6.40	8.20	13.80
ecoli2	**7.80**	8.20	8.80	16.00	5.00	5.20	**4.20**	8.60
new-thyroid1	**3.40**	**3.40**	**3.40**	4.40	**2.80**	**2.80**	3.00	4.00
new-thyroid2	**3.60**	**3.60**	**3.60**	6.20	**3.00**	**3.00**	3.60	4.60
ecoli1	**5.20**	6.60	6.80	9.00	4.20	4.00	**3.40**	5.80
vehicle0	**13.60**	**13.60**	17.80	29.60	**8.52**	9.20	11.20	26.20
glass-0-1-2-3_vs_4-5-6	**4.40**	**4.40**	5.80	8.60	**4.60**	4.80	4.80	10.20
haberman	5.40	5.60	**5.00**	10.80	2.60	2.60	**2.20**	2.80
vehicle1	**16.80**	17.80	20.60	70.60	7.48	**6.68**	9.24	28.40
vehicle2	**9.60**	**9.60**	11.80	18.60	**9.00**	9.40	10.64	16.60
vehicle3	18.00	**17.60**	27.40	76.40	**8.12**	8.60	11.28	26.40
yeast1	**13.92**	14.20	17.60	55.40	**3.80**	4.60	6.60	11.60
glass0	5.80	**5.60**	**5.60**	13.00	2.80	2.80	**2.20**	5.00
iris0	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**
pima	9.80	**8.80**	10.00	30.00	4.60	**4.40**	4.80	8.80
ecoli0_vs_1	**4.00**	**4.00**	**4.00**	4.20	**2.00**	**2.00**	**2.00**	2.20
wisconsin	**9.60**	**9.60**	12.40	35.20	**6.40**	7.20	**6.40**	14.00
glass1	**6.60**	**6.60**	7.20	14.00	**3.56**	**3.56**	3.80	6.80
Mean	**9.44**	9.47	11.53	25.54	**6.52**	6.81	7.84	17.78
Median	**7.20**	7.40	8.00	13.80	**4.60**	4.80	5.40	10.20

Table I.9: Results for *Number of Rules* over imbalanced datasets preprocessed with SMOTE.

dataset	C4.5-based				CHAID*-based			
	UnPART	BFPART	PART	C4.5	UnPART	BFPART	PART	CHAID*
nursery	**1.93**	2.04	2.27	4.10	2.19	**2.18**	3.23	4.35
abalone	4.83	5.03	**3.98**	7.67	**3.62**	4.03	4.30	5.02
ecoli	2.84	2.57	**2.46**	5.36	**4.02**	4.11	4.56	4.69
lymphography	2.10	**2.05**	2.34	3.68	**2.30**	2.52	2.46	2.99
car	2.36	2.38	**2.30**	4.62	**2.31**	2.35	2.98	5.17
zoo	1.55	1.55	**1.25**	3.76	1.77	**1.76**	**1.76**	2.51
flare	1.64	**1.57**	2.68	3.33	**1.36**	2.61	3.06	1.77
glass	2.97	3.02	**2.78**	5.55	**2.93**	**2.93**	3.10	3.88
cleveland	4.03	3.94	**3.77**	6.61	2.06	**2.00**	2.75	3.50
dermatology	**2.38**	**2.38**	**2.38**	3.90	**2.59**	3.09	3.43	4.71
balance	**2.18**	**2.18**	2.98	6.06	**1.89**	2.04	2.69	4.00
penbased	3.82	3.84	**3.31**	6.86	**6.44**	6.50	6.90	7.39
newthyroid	**1.58**	1.62	1.69	3.42	**2.00**	2.16	2.16	3.17
hepatitis	**1.28**	**1.28**	1.45	2.48	**.99**	1.09	1.25	1.88
contraceptive	6.06	**4.88**	5.14	10.14	**2.80**	2.98	3.63	4.57
vehicle	4.45	4.42	**3.47**	7.62	**3.88**	4.02	4.61	6.19
haberman	**.63**	**.63**	1.23	1.26	**.20**	**.20**	.30	.43
wine	1.29	1.29	**1.14**	2.45	**2.05**	**2.05**	2.25	3.13
breast	**.80**	**.80**	1.94	2.11	**.79**	**.79**	1.25	2.24
german	2.48	**2.30**	2.68	5.27	1.41	**1.39**	2.46	3.56
iris	.95	.95	**.92**	2.35	**1.02**	1.20	1.20	2.36
wisconsin	**.93**	**.93**	.98	1.78	**.99**	1.30	2.00	1.93
tictactoe	2.31	**2.30**	2.70	4.49	**2.13**	2.13	2.82	4.94
pima	**1.90**	**1.90**	1.93	5.55	1.65	**1.63**	2.05	3.15
magic	2.32	**2.31**	2.68	6.94	**2.32**	2.48	3.61	3.60
bupa	**2.00**	2.18	2.22	5.74	**.93**	**.93**	1.42	2.16
heart	**1.56**	1.59	2.48	3.74	**1.26**	1.34	2.02	3.17
australian	**2.09**	2.24	3.47	6.27	**1.05**	1.89	2.65	2.20
crx	1.58	**1.53**	2.41	4.54	**1.07**	1.61	2.71	2.36
ring	3.58	3.58	**2.61**	8.64	**7.02**	8.14	8.14	7.95
Mean	2.35	**2.31**	2.46	4.88	**2.23**	2.45	2.93	3.63
Median	**2.10**	2.18	2.44	4.58	**2.03**	2.09	2.70	3.34

Table I.10: Results for *Length* over standard datasets.

dataset	C4.5-based				CHAID*-based			
	UnPART	BFPART	PART	C4.5	UnPART	BFPART	PART	CHAID*
abalone19		**.00**	**.00**	**.00**	**.00**	**.00**	.30	**.00**
yeast6	**.89**	**.89**	1.17	2.77	**.15**	**.15**	.28	.45
yeast5	**1.17**	1.25	1.39	4.65	**.95**	1.00	1.20	2.50
yeast4	**1.13**	**1.13**	1.71	3.81	**.68**	**.68**	1.14	1.90
yeast2_vs_8	**.20**	.60	.73	.65	**.90**	**.90**	1.17	1.53
glass5	**.75**	**.75**	**.75**	2.25	**.67**	**.67**	**.67**	1.67
abalone9_vs_18	1.09	**1.07**	1.62	3.06	**.83**	**.83**	1.17	1.50
glass4	**1.17**	**1.17**	**1.17**	2.73	**.80**	**.80**	**.80**	1.69
ecoli4	**.87**	**.87**	1.17	1.93	**.97**	1.02	1.73	2.11
glass2	**1.18**	**1.18**	1.34	3.51	**.10**	**.10**	**.10**	.20
vowel0	1.48	1.59	**1.39**	3.20	**2.50**	2.75	2.71	4.29
page-blocks0	3.49	3.58	**3.40**	6.68	**3.61**	4.17	5.60	5.86
ecoli3	**1.00**	**1.00**	1.05	2.96	**.73**	**.73**	**.73**	1.73
yeast3	**1.50**	**1.50**	2.04	4.39	**.93**	1.00	1.48	2.12
glass6	**.80**	**.80**	1.07	1.81	**.87**	**.87**	**.87**	1.92
segment0	1.72	1.72	**1.72**	3.70	**1.55**	1.59	2.40	4.06
ecoli2	**1.35**	**1.35**	1.36	3.56	**1.20**	**1.20**	1.90	2.69
new-thyroid1	**1.27**	**1.27**	**1.27**	2.80	**1.02**	**1.02**	**1.02**	2.37
new-thyroid2	**.97**	1.04	1.04	2.23	**.93**	**.93**	**.93**	2.25
ecoli1	**1.06**	1.09	1.37	3.31	**1.33**	**1.33**	1.47	2.32
vehicle0	**2.21**	2.39	2.31	6.83	2.33	**2.33**	2.76	4.87
glass-0-1-2-3_vs_4-5-6	**1.23**	**1.23**	1.52	2.80	1.36	**1.35**	1.47	2.37
haberman	**.63**	**.63**	1.23	1.26	**.20**	**.20**	.30	.40
vehicle1	3.20	4.01	**3.05**	6.93	**1.16**	1.24	2.06	2.64
vehicle2	**2.05**	2.26	2.22	5.00	2.49	**2.49**	2.99	4.45
vehicle3	4.05	4.06	**2.64**	8.34	**1.76**	1.92	2.76	3.22
yeast1	2.18	**2.09**	2.52	6.29	**1.52**	1.60	2.50	2.90
glass0	1.75	1.85	**1.74**	4.06	**1.01**	**1.01**	1.31	2.03
iris0	**.50**	**.50**	**.50**	1.00	**.50**	**.50**	**.50**	1.00
pima	1.95	**1.90**	1.93	5.55	1.66	**1.63**	2.05	3.15
ecoli0_vs_1	**.50**	**.50**	.70	1.00	**.50**	**.50**	1.00	1.00
wisconsin	**1.03**	1.05	1.07	1.62	**1.28**	1.40	1.97	2.79
glass1	**1.90**	1.91	2.10	4.26	**1.13**	1.20	1.20	2.37
Mean	**1.40**	1.46	1.52	3.48	**1.14**	1.19	1.53	2.31
Median	**1.17**	1.18	1.37	3.20	**.97**	1.01	1.20	2.25

Table I.11: Results for *Length* over imbalanced datasets.

dataset	C4.5-based				CHAID*-based			
	UnPART	BFPART	PART	C4.5	UnPART	BFPART	PART	CHAID*
abalone19	4.92	4.29	**3.63**	9.73	**5.40**	6.40	7.52	9.78
yeast6	2.80	2.89	**2.55**	5.69	**4.34**	4.71	6.25	6.98
yeast5	**1.93**	1.95	1.97	4.77	**2.99**	4.22	4.30	4.99
yeast4	3.97	4.20	**3.21**	7.42	**4.83**	4.92	5.04	7.58
yeast2_vs_8	**2.09**	2.17	2.09	5.80	**2.76**	2.81	3.24	5.45
glass5	1.90	1.90	**1.70**	3.40	**2.24**	2.34	2.64	3.79
abalone9_vs_18	3.54	3.84	**2.88**	7.48	**2.97**	3.05	3.52	5.71
glass4	**1.83**	**1.83**	**1.83**	3.99	**2.08**	2.75	2.80	3.26
ecoli4	**1.61**	**1.61**	1.65	4.10	**1.85**	2.18	2.17	3.58
glass2	**2.29**	**2.29**	2.46	6.68	**2.54**	2.60	2.93	4.53
vowel0	**2.10**	2.23	2.16	4.00	**3.34**	3.63	3.93	4.45
page-blocks0	5.08	5.04	**3.76**	9.40	**5.03**	5.36	6.25	8.01
ecoli3	**1.80**	**1.80**	1.86	5.04	2.20	2.33	**2.05**	3.86
yeast3	3.09	3.07	**2.74**	6.44	**2.94**	3.23	3.67	5.55
glass6	**1.65**	**1.65**	1.85	3.33	1.94	1.94	**1.74**	2.95
segment0	2.07	2.04	**2.01**	4.32	**2.19**	2.40	3.19	5.20
ecoli2	1.90	**1.76**	1.84	5.22	**1.99**	2.43	2.80	3.76
new-thyroid1	**1.16**	**1.16**	**1.16**	2.41	**1.13**	**1.13**	1.43	2.03
new-thyroid2	1.40	1.40	**1.20**	2.73	**1.27**	1.33	1.43	2.24
ecoli1	1.33	**1.31**	1.45	3.69	1.27	**1.17**	1.68	2.60
vehicle0	2.81	2.88	**2.55**	7.21	**3.13**	3.26	3.92	6.14
glass-0-1-2-3_vs_4-5-6	**1.47**	**1.47**	1.55	3.57	**1.94**	2.02	2.05	3.61
haberman	**1.75**	2.02	2.18	4.06	**.68**	**.68**	.87	1.52
vehicle1	5.38	5.75	**3.69**	8.56	**2.63**	2.76	3.88	5.41
vehicle2	2.49	**2.49**	2.71	5.11	**2.83**	3.10	3.85	4.81
vehicle3	4.79	5.11	**3.02**	10.30	**2.89**	3.05	4.05	5.48
yeast1	3.27	2.97	**2.79**	7.70	1.98	**1.85**	3.08	3.90
glass0	**1.96**	2.11	2.11	4.59	**1.60**	**1.60**	1.90	2.72
iris0	**.50**	**.50**	**.50**	1.00	**.50**	**.50**	**.50**	1.00
pima	2.60	**2.37**	2.67	6.16	**1.70**	1.72	1.94	3.39
ecoli0_vs_1	**.76**	**.76**	**.76**	2.33	**.60**	**.60**	1.00	1.13
wisconsin	**.98**	**.98**	1.05	1.82	**1.48**	1.59	2.12	2.68
glass1	**2.02**	2.05	2.05	4.91	**1.58**	**1.58**	1.75	2.93
Mean	2.40	2.42	**2.17**	5.24	**2.39**	2.58	3.02	4.27
Median	**2.02**	2.05	2.09	4.91	**2.19**	2.40	2.80	3.86

Table I.12: Results for *Length* over imbalanced datasets preprocessed with SMOTE.

dataset	C4.5-based				CHAID*-based			
	UnPART	BFPART	PART	C4.5	UnPART	BFPART	PART	CHAID*
nursery	107	110	55	**14**	633	576	226	**94**
abalone	1867	1623	1858	**139**	794	871	849	**575**
ecoli	50	40	34	**19**	208	204	174	**91**
lymphography	14	19	16	**8**	94	99	59	**46**
car	145	198	47	**18**	708	703	330	**114**
zoo	9	8	8	**7**	60	99	31	**24**
flare	56	51	47	**20**	175	306	184	**84**
glass	70	64	52	**21**	321	299	265	**148**
cleveland	107	99	82	**26**	128	143	122	**89**
dermatology	23	18	18	**13**	381	578	243	**117**
balance	47	47	27	**13**	57	58	39	**27**
penbased	779	693	371	**106**	13565	11505	5144	**1420**
newthyroid	11	**6**	7	7	69	55	47	**44**
hepatitis	8	**2**	7	7	22	18	**16**	24
contraceptive	657	710	314	**71**	254	249	228	**106**
vehicle	390	370	227	**68**	3567	3162	1930	**847**
haberman	**4**	7	6	5	**6**	7	7	11
wine	16	17	16	**11**	243	175	146	**123**
breast	11	9	22	**9**	42	41	41	**25**
german	204	193	284	**51**	1071	1157	1173	**462**
iris	10	**5**	5	6	22	22	17	**16**
wisconsin	18	14	16	**7**	160	179	177	**77**
tictactoe	89	90	54	**15**	319	267	129	**94**
pima	67	59	53	**33**	555	596	515	**338**
magic	1661	1432	1914	**886**	21053	21710	25879	**12293**
bupa	26	31	16	**13**	45	47	44	**33**
heart	21	24	28	**13**	106	106	137	**81**
australian	79	74	97	**33**	417	499	443	**260**
crx	50	48	128	**27**	357	432	529	**257**
ring	965	821	1294	**586**	8835	8842	8526	**8329**
Mean	252	229	237	**75**	1809	1767	1588	**875**
Median	53	49	47	**16**	248	258	180	**94**

Table I.13: Results for Time (in milliseconds) over standard datasets.

dataset	C4.5-based				CHAID*-based			
	UnPART	BFPART	PART	C4.5	UnPART	BFPART	PART	CHAID*
abalone19	343	293	295	**287**	6623	**6210**	7475	6270
yeast6	18	17	16	**11**	213	198	**194**	206
yeast5	21	19	15	**12**	273	**202**	205	212
yeast4	19	21	20	**14**	303	277	**248**	250
yeast2_vs_8	**5**	6	7	7	57	**50**	59	52
glass5	**8**	8	8	8	66	**42**	52	57
abalone9_vs_18	51	46	54	**40**	731	693	734	**608**
glass4	11	9	**8**	11	104	73	**73**	93
ecoli4	8	**6**	7	8	71	**50**	62	63
glass2	19	15	15	**13**	29	26	**24**	29
vowel0	179	154	151	**139**	7295	5876	**5529**	5664
page-blocks0	1058	914	934	**596**	33853	33153	42501	**22628**
ecoli3	9	9	9	**8**	58	45	42	**40**
yeast3	28	23	27	**16**	303	272	301	**246**
glass6	10	10	10	**9**	108	87	**83**	90
segment0	781	663	789	**551**	19787	14595	24362	**12074**
ecoli2	10	11	10	**7**	66	67	54	**46**
new-thyroid1	8	7	11	**7**	35	31	27	**26**
new-thyroid2	**6**	6	6	**6**	34	20	**17**	25
ecoli1	11	11	**10**	10	79	84	82	**62**
vehicle0	81	74	51	**31**	1201	1046	900	**456**
glass-0-1-2-3_vs_4-5-6	13	12	13	**10**	125	110	120	**90**
haberman	14	6	**6**	8	11	27	28	**11**
vehicle1	189	223	129	**49**	709	714	718	**420**
vehicle2	69	76	51	**29**	1287	1040	950	**506**
vehicle3	222	178	108	**56**	903	828	912	**438**
yeast1	63	61	44	**30**	373	381	508	**256**
glass0	21	22	18	**14**	104	120	97	**75**
iris0	4	3	3	**3**	9	4	**4**	11
pima	68	65	55	**35**	556	575	532	**336**
ecoli0_vs_1	5	**4**	5	5	30	24	**19**	27
wisconsin	18	14	16	**8**	200	193	173	**91**
glass1	21	21	18	**13**	118	100	89	**88**
Mean	103	91	88	**62**	2294	2037	2642	**1562**
Median	19	17	16	**12**	125	120	120	**91**

Table I.14: Results for Time (in milliseconds) over imbalanced datasets.

dataset	C4.5-based				CHAID*-based			
	UnPART	BFPART	PART	C4.5	UnPART	BFPART	PART	CHAID*
abalone19	50138	38491	29208	**7766**	1411243	1415336	1086926	**252893**
yeast6	691	587	647	**403**	45184	38775	44531	**13834**
yeast5	854	731	724	**467**	41557	36696	28528	**12344**
yeast4	1473	1298	782	**334**	62353	57358	49315	**15453**
yeast2_vs_8	120	101	77	**50**	3902	3155	3113	**1296**
glass5	33	**27**	28	34	674	492	556	**462**
abalone9_vs_18	1505	1327	985	**220**	18950	17504	15040	**4769**
glass4	40	33	48	**32**	756	595	470	**419**
ecoli4	54	45	52	**33**	924	704	900	**552**
glass2	93	79	81	**50**	698	592	910	**484**
vowel0	1210	1024	907	**622**	68271	54730	44627	**19085**
page-blocks0	13534	11673	7944	**2798**	776982	741953	763009	**145189**
ecoli3	60	48	39	**32**	676	575	535	**422**
yeast3	565	491	587	**320**	25298	22970	21088	**11619**
glass6	32	30	**21**	28	489	378	**304**	325
segment0	4480	3765	4726	**3269**	148637	115337	170759	**96426**
ecoli2	45	41	39	**33**	674	597	562	**423**
new-thyroid1	10	10	**9**	15	140	**107**	110	119
new-thyroid2	13	**10**	12	15	172	120	**112**	121
ecoli1	25	30	26	**25**	361	332	308	**254**
vehicle0	164	164	99	**54**	2694	2300	1818	**991**
glass-0-1-2-3_vs_4-5-6	22	22	**21**	23	475	389	354	**292**
haberman	12	19	**7**	17	15	**13**	18	23
vehicle1	551	535	256	**93**	2301	2113	2155	**957**
vehicle2	116	105	84	**52**	2772	2258	1823	**853**
vehicle3	557	547	264	**102**	2538	2513	2525	**956**
yeast1	402	367	446	**158**	5653	6328	7564	**3550**
glass0	33	30	**25**	28	250	222	201	**192**
iris0	4	**2**	4	11	17	**9**	**9**	21
pima	152	135	121	**63**	1303	1269	1048	**654**
ecoli0_vs_1	8	**5**	7	12	67	31	**31**	52
wisconsin	22	27	23	**15**	313	636	258	**109**
glass1	32	28	32	**24**	327	260	230	**176**
Mean	2335	1873	1465	**521**	79596	76565	68174	**17737**
Median	93	79	77	**50**	924	704	910	**552**

Table I.15: Results for Time (in milliseconds) over imbalanced datasets preprocessed with SMOTE.